Filesystem Hierarchy Standard — Draft 9

Filesystem Hierarchy Standard Group

edited by Daniel Quinlan

ABSTRACT

This standard consists of a set of requirements and guidelines for file and directory placement under UNIX-like operating systems. The guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.

September 30, 1996

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

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

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, 1996 Daniel Quinlan

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

Permission is granted to copy and distribute modified versions of this standard under the conditions for verbatim copying, provided also that the title page is labeled as modified including a reference to the original standard, provided that information on retrieving the original standard is included, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this standard into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the copyright holder.

1. Introduction

1.1 Status of the Standard

This proposed standard is currently being distributed for review and comment.

Comments on this draft are welcome from any interested party. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

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

1.2 Organization of the Standard

This standard is divided into these sections:

1. Introduction
2. The Filesystem: a statement of some guiding principles.
3. The Root Directory.
4. The `/usr` Hierarchy.
5. The `/var` Hierarchy.
6. Operating System Specific Annex.

1.3 Conventions

A constant-width font is used for displaying 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 "`*`".

1.4 Background of the FHS

The process of developing a standard filesystem hierarchy began in August 1993 with an effort to restructure the file and directory structure of Linux. The FSSTND, a filesystem hierarchy standard specific to the Linux operating system, was released on February 14, 1994. Subsequent revisions were released on October 9, 1994 and March 28, 1995.

In early 1995, the goal of developing a more comprehensive version of FSSTND to address not only Linux, but other UNIX-like systems was adopted in conjunction with members of the BSD development community. As a result, a concerted effort was made to focus on issues that were general to UNIX-like systems. In recognition of this widening of scope, the name of the standard (and the mailing list) was changed to Filesystem Hierarchy Standard or FHS for short.

Volunteers who have contributed extensively to this standard are listed at the end of this document. This standard represents a consensus view of those and other contributors.

1.5 Scope

This document specifies a standard filesystem hierarchy for FHS filesystems by specifying the location of files and directories, and the contents of some system files.

This standard has been designed to be used by system integrators, package developers, and system administrators in the construction and maintenance of FHS compliant filesystems. It is primarily intended to be a reference and is not a tutorial on how to manage a conforming filesystem hierarchy.

The FHS grew out of earlier work on FSSTND, a filesystem organization standard for the Linux operating system. It builds on FSSTND to address interoperability issues not just in the Linux community but in a wider arena including 4.4BSD-based operating systems. It incorporates lessons learned in the BSD world and elsewhere about multi-architecture support and the demands of heterogeneous networking.

Although this standard is more comprehensive and complete than any other previous attempt at file hierarchy standardization, updates may become periodically become necessary. The requirements of the 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 FHS group plans to release supplementary drafts in addition to periodic updates to this document. However, a specific goal is backwards compatibility from one release of this document to the next.

Comments related to this standard are welcomed by the FHS group. Any comments or suggestions for changes should be directed to the FHS editor (Daniel Quinlan <quinlan@pathname.com>), or if you prefer, one of the listed contributors. Typographical or grammatical comments should be directed to the FHS editor.

Please do not send mail to the mailing list without first contacting the FHS editor 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 FHS editor. Since this standard represents a 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.

1.6 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 consequent upon moving away from the former de-facto standards.
- Gain approval of distributors, developers, and other important people in different software communities (such as Linux and BSD), as well as encouraging them to give us their suggestions.
- Provide a standard that different UNIX-like systems will choose to follow because it solves the problems above and provides the most sensible structure for the filesystems of UNIX-like systems.

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

1.7 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 FHS editor. It is anticipated that some variation will be tolerated in borderline cases.

To qualify as partially FHS compliant or partially FHS compatible an implementation must provide a list of all places at which it and the FHS 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 FHS mailing list or the FHS editor.

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 minix filesystem is not supported by a distribution, the minix 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 assumes that the operating system underlying an FHS-compliant file system supports the same basic security features found in most UNIX filesystems. 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 and BSD communities
- The implementation of other filesystem structures
- Applicable standards

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

Shareable data is that which can be shared between several different hosts; unshareable is that which must be specific to a particular host. 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 hosts to save space and ease the task of maintenance.
- In a networked environment, certain files contain information specific to a single host. Therefore these filesystems cannot be shared (without taking special measures).
- Historical implementations of Linux and BSD filesystems 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 FHS-compliant systems, using the postal mail system as a ‘‘network’’.

The "static" versus "variable" distinction affects the filesystem in two 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 /opt	/etc /boot
variable	/var/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 on the root partition to mount other filesystems. This includes utilities, configuration, boot loader information, and other essential start-up data. `/usr`, `/opt`, and `/var` are designed such that they may be located on other filesystems.
- 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 occasionally mounted from very small media.
- 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.

Software should never create or require special files or subdirectories in the root directory. Other locations in the FHS hierarchy provide more than enough flexibility for any package.

BEGIN RATIONALE

There are several reasons why using a root subdirectory is prohibited.

- It demands space on a root partition which the system administrator may want kept small and simple for either performance or security reasons.
- It evades whatever discipline the system administrator may have set up for distributing standard file hierarchies across mountable volumes.

END RATIONALE

/ — the root directory

bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
home	User home directories
lib	Shared libraries
mnt	Mount point of temporary partitions
opt	Add-on application software packages
proc	Process information pseudo-filesystem
root	Home directory for the root user
sbin	Essential system binaries
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data

Each directory listed above is specified in detail in separate subsections below. /usr and /var each have a complete section in this document due to the complexity of those directories.

The operating system kernel image should be located in either / or /boot. 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.

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.

  { 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`.

Note: The `[` *and* `test` *commands are built into most commonly used Bourne shell (*`/bin/sh`*) replacements. These two commands do not have to be placed in* `/bin`*; they may be placed in* `/usr/bin`*. They must be included as separate binaries with any UNIX or UNIX-like system attempting to comply with the POSIX.2 standard.*

- `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 an ftp connection) should be available on the root partition.

  Restoration commands may appear in either `/bin` or `/usr/bin` on different 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 required for the boot process except configuration files and the map installer. Thus `/boot` stores data which is used before the kernel beings executing user-mode programs. This may include saved master boot sectors, sector map files, and other data that is not directly edited by hand. Programs necessary to arrange for the boot loader to be able to boot a file should be placed in `/sbin`. Configuration files for boot loaders should be placed in `/etc`.

The operating system kernel should be located either in `/` or in `/boot`.

Note: On some i386 machines, it may be necessary for `/boot` *to be located on a separate partition below cylinder 1024 of the boot device due to hardware constraints.*

3.3 /dev : Device files

The /dev directory is the location of special or device files.

If it is possible that devices in /dev will need to be manually created, /dev shall contain a command named MAKEDEV, which can create devices as needed. It may also contain a MAKEDEV.local for any local devices.

If required, MAKEDEV should have provisions for creating any device that may be found on the system, not just those that a particular implementation installs.

3.4 /etc : Host-specific system configuration

/etc contains configuration files and directories, which are specific 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, /usr/sbin, or possibly /usr/libexec. 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 — Host-specific system configuration

X11	Configuration for the X Window System
opt	Configuration for /opt

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:

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

- Networking files:

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

Notes:

There are several different models for setup of the "rc" command scripts which are invoked at boot time such as the /etc/rc. BSD model and the System V model. No specific model is required, but a specification of this area may be added to a future version of this standard.*

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

3.4.1 /etc/X11 : Configuration for the X Window System

`/etc/X11` is the recommended location for all X11 host-specific 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`. Some local variable data for `xdm` is stored in `/var/state/xdm`.

3.4.2 /etc/opt : Configuration files for /opt

Host-specific configuration files for add-on application software packages shall be installed within the directory `/etc/opt/<package>`, where `<package>` is the name of the subtree in `/opt` where the static data from that package is stored. No structure is imposed on the internal arrangement of `/etc/opt/<package>`.

If a configuration file must reside in a different location in order for the package or system to function properly, it may be placed in a location other than `/etc/opt/<package>`.

BEGIN RATIONALE

Refer to the rationale for `/opt`.

END RATIONALE

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 host to host. This section describes only a suggested placement for user home directories; nevertheless we recommend that all FHS-compliant 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 hosts 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 /opt : Add-on application software packages

/opt — Add-on application software packages

`<package>`	Static package objects

`/opt` is reserved for the installation of add-on application software packages.

A package to be installed in `/opt` shall locate its static files in a separate `/opt/<package>` directory tree, where `<package>` is a name that describes the software package.

Programs to be invoked by users shall be located in the directory `/opt/<package>/bin`. If the package includes UNIX manual pages, they shall be located in `/opt/<package>/man` and the same substructure as `/usr/share/man` shall be used.

The directories `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib`, and `/opt/man` are reserved for local system administrator use. Packages may provide "front-end" files intended to be placed in (by linking or copying) these reserved directories by the local system administrator, but shall function normally in the absence of these reserved directories.

Package files that are variable (change in normal operation) should be installed in `/var/opt`. See the section on `/var/opt` for more information.

Host-specific configuration files should be installed in `/etc/opt`. See the section on `/etc` for more information.

No other package files should exist outside the `/opt`, `/var/opt`, and `/etc/opt` hierarchies except for those package files that must reside in specific locations within the filesystem tree in order to function properly. For example, device lock files must be placed in `/var/lock` and devices must be located in `/dev`.

BEGIN RATIONALE

The use of `/opt` for add-on software is a well-established practice in the UNIX community. The System V Application Binary Interface [AT&T 1990], based on the System V Interface Definition (Third Edition), provides for an `/opt` structure very similar to the one defined here.

The Intel Binary Compatibility Standard v. 2 (iBCS2) also provides a similar structure for `/opt`.

Generally, all data required to install a package on a system should be present within `/opt/<package>`, including files intended to be copied into `/etc/opt/<package>` and `/var/opt/<package>` as well as reserved directories in `/opt`.

END RATIONALE

3.9 /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.

3.10 /root : Home directory for the root user (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, and that it be used solely for system administration. For this reason, we recommend that subdirectories such as `Mail` and `News` not appear in the root account's home directory, and that mail for administration roles such as root, postmaster and webmaster be forwarded to an appropriate user.

3.11 /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 (non-root) user will ever 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 or more of `ext, ext2, minix, msdos, xia` and perhaps others

- Networking commands:

  `{ ifconfig, route }`

3.12 /tmp : Temporary files

The `/tmp` directory shall be made available for programs that require temporary files.

Although data stored in `/tmp` may be deleted in a site-specific manner, it is recommended that files and directories located in `/tmp` be deleted whenever the system is booted.

Programs shall not assume that any files or directories in `/tmp` are preserved between invocations of the program.

BEGIN RATIONALE

IEEE standard P1003.2 (POSIX, part 2) makes requirements that are similar to the above section.

FHS added the recommendation that `/tmp` be cleaned at boot time on the basis of historical precedent and common practice, but did not make it a requirement because system administration is not within the scope of this standard.

END RATIONALE

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 hosts running FHS-compliant and should not be written to. Any information that is host-specific or varies with time is stored elsewhere.

No large software packages should use a direct subdirectory under the `/usr` hierarchy. An exception is made for the X Window System because of considerable precedent and widely-accepted practice. This section of the standard specifies the location for most such packages.

/usr — Secondary Hierarchy

`X11R6`	X Window System, version 11 release 6
`X386`	X Window System, version 11 release 5 on x86 platforms
`bin`	Most user commands
`games`	Games and educational binaries
`include`	Header files included by C programs
`lib`	Libraries
`libexec`	Internal binaries
`local`	Local hierarchy (empty after main installation)
`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/log
/usr/preserve -> /var/state/<editor>
/usr/spool -> /var/spool
/usr/tmp -> /var/tmp
/usr/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 `elvis` uses 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.

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.

Host-specific data in `/usr/X11R6/lib/X11` should be interpreted as a demonstration file. Applications requiring informmation about the current host must reference a configuration file in `/etc/X11`, which may be linked to a file in `/usr/X11R6/lib`.

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 their locations. 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/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
bsd	BSD compatibility include files
g++	GNU C++ include files
gnu	GNU include files
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.

4.5 /usr/lib : Libraries for programming

/usr/lib is intended for object files and libraries.

Historically, /usr/lib has also included compiler program binaries and static data of various kinds — both executable code (for example, GCC's internal binaries may be located under /usr/lib/gcc-lib) and other types of data.

Miscellaneous application-specific static files and subdirectories should be placed in /usr/share, and application-specific binaries in /usr/libexec (for internal binaries), /usr/bin (for user commands), or /usr/sbin (for system commands). However, distributors may place application-specific static data and binaries in subdirectories of /usr/lib instead, at their discretion. Such application-specific directories might include, for example, emacs, gcc-lib, mf, news, smail, texmf, or uucp.

Some executable commands such as makewhatis and sendmail have also been traditionally placed in /usr/lib. makewhatis is an internal binary and should be placed in a binary directory; users access

only catman. Newer sendmail binaries are now placed by default in /usr/sbin; a symbolic link should remain from /usr/lib. Additionally, systems using Smail should place Smail in /usr/sbin/smail, and /usr/sbin/sendmail should be a symbolic link to it.

A symbolic link /usr/lib/X11 pointing to the lib/X11 directory of the default X distribution is required if X is installed. Note: No host-specific data for the X Window System should be stored in /usr/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 (probably in /usr/X11R6/lib/X11).

4.6 /usr/libexec : Internal binaries

/usr/libexec is for executable programs to be run by other programs rather than by users from the shell. The name is derived from "library executables".

Examples of programs located in /usr/libexec include internet daemons started by inetd; programs started by init such as getty; and compiler binaries started by the C compiler such as cc1.

Applications may use a single subdirectory under /usr/libexec. If an application uses a subdirectory, all programs exclusively used by the application should be placed within that subdirectory. Examples of subdirectories located in /usr/libexec include uucp for UUCP component binaries and lpr for printer daemon filter programs.

No specific programs must be placed here, but it is recommended that any binaries in /usr/lib and /usr/sbin that meet the above criteria be moved to this directory.

BEGIN RATIONALE

The concept of a libexec directory was introduced in the Net/2 distribution of the BSD operating system. It was created to remove executables that weren't intended to be executed directly by users from the standard execution path. The GNU Coding Standards also use a libexec directory.

A similar concept has been used in other UNIX operating systems such as OSF/1, which instead uses the directory /usr/lbin for "back-end executables". The name /usr/lbin was not chosen because it was first used for local binaries (rather than internal binaries) in System III. We also felt that /usr/lbin would be more easily mistaken for a directory that was meant to be placed in the execution path.

Additionally, as internal binaries migrate from /usr/lib to /usr/libexec, /usr/lib, originally designed for object libraries and archives, should regain some measure of clarity.

END RATIONALE

4.7 /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 hosts, but not found in /usr.

/usr/local — Local hierarchy

`bin`	Local binaries
`etc`	Local configuration
`games`	Local game binaries
`lib`	Local libraries
`libexec`	Local internal binaries
`share`	Local architecture-independent hierarchy
`src`	Local source code

This directory should always be empty after first installing a FHS-compliant system. 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.

4.8 /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 hosts (e.g., NFS exports).

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

4.9 /usr/share : Architecture-independent data

/usr/share — Architecture-independent data

dict	Word lists
doc	Miscellaneous documentation
games	Static data files for /usr/games
info	GNU Info system's primary directory
locale	Locale information
man	Online manuals
nls	Native language support
misc	Miscellaneous architecture-independent data
terminfo	Directories for terminfo database
tmac	troff macros not distributed with groff
zoneinfo	Timezone information and configuration

The /usr/share hierarchy is for all read-only architecture independent data files. Much of this data originally lived in /usr (man, doc) or /usr/lib (dict, terminfo, zoneinfo). This hierarchy is intended to be shareable among all architecture platforms of a given OS; thus, for example, a site with i386, Alpha, and PPC platforms might maintain a single /usr/share directory that is centrally-mounted. Note, however, that /usr/share is generally not intended to be shared by different OSes or by different releases of the same OS.

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

Note that Linux currently uses DBM-format database files. While these are not architecture-independent, they are allowed in /usr/share as a short-term measure pending an anticipated switch to the DB 2.0 format. DB 2.0 uses an architecture independent format.

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/games.

It is recommended that application-specific, architecture-independent directories be placed here. Such directories include groff, perl, ghostscript, texmf, and kbd (Linux) or syscons (BSD). They may, however, be placed in /usr/lib for backwards compatibility, at the distributor's discretion. Similarly, a /usr/lib/games hierarchy may be used in addition to the /usr/share/games hierarchy if the distributor wishes to place some game data there.

4.9.1 /usr/share/dict : Word lists

Recommended files for /usr/share/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/share/dict/american-english or

/usr/share/dict/british-english.

Word lists for other languages may be added using the English name for that language, e.g., /usr/share/dict/french, /usr/share/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.

BEGIN RATIONALE

The reason that only word lists are located here is that they are the only files common to all spell checkers.

END RATIONALE

4.9.2 /usr/share/man : Manual pages

This section details the organization for manual pages throughout the system, including /usr/share/man. Also refer to the section on /var/cache/man.

Manual pages are stored in <mandir>/<locale>/man[1-9]/<arch>.
An explanation of <mandir>, <locale>, and <arch>
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/share/man. /usr/share/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/share/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/share/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/share/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/share/man/en
English	United Kingdom	ASCII	/usr/share/man/en_GB
English	United States	ASCII	/usr/share/man/en_US
French	Canada	ISO 8859-1	/usr/share/man/fr_CA
French	France	ISO 8859-1	/usr/share/man/fr_FR
German	Germany	ISO 646	/usr/share/man/de_DE.646
German	Germany	ISO 6937	/usr/share/man/de_DE.6937
German	Germany	ISO 8859-1	/usr/share/man/de_DE.88591
German	Switzerland	ISO 646	/usr/share/man/de_CH.646
Japanese	Japan	JIS	/usr/share/man/ja_JP.jis
Japanese	Japan	SJIS	/usr/share/man/ja_JP.sjis
Japanese	Japan	UJIS (or EUC-J)	/usr/share/man/ja_JP.ujis

Similarly, provision must be made for manual pages which are architecture-dependent, such as device-driver documentation or low-level system administration commands. These should be placed under an <arch> directory in the appropriate man[1-9] directory; for example, a man page for the i386 ctrlaltdel(8) command might be placed in /usr/share/man/<locale>/man8/i386/ctlaltdel.8.

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

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/share/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/share/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/cache/man 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 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 systems that document the kernel source code.

4.9.3 /usr/share/misc : Miscellaneous architecture-independent data

This directory contains miscellaneous architecture-independent files which don't require a separate subdirectory under `/usr/share`. It is a required directory under `/usr/share`.

The following files, if present, should be located under `/usr/share/misc`:

```
{ ascii, magic, termcap, termcap.db }
```

Other (application-specific) files may appear here, but a distributor may place them in `/usr/lib` at their discretion. Some such files include:

```
{ airport, birthtoken, eqnchar, getopt, gprof.callg, gprof.flat,
  inter.phone, ipfw.samp.filters, ipfw.samp.scripts, keycap.pcvt,
  mail.help, mail.tildehelp, man.template, map3270, mdoc.template,
  more.help, na.phone, nslookup.help, operator, scsi_modes, sendmail.hf,
  style, units.lib, vgrindefs, vgrindefs.db, zipcodes }
```

4.10 /usr/src : Source code

Any non-local source code should be placed in this subdirectory.

5. The /var Hierarchy

/var — Variable data

`account`	Process accounting logs (if supported)
`cache`	Application cache data
`crash`	System crash dumps (if supported)
`cron`	cron and at jobs
`games`	Variable game data
`lock`	Lock files
`log`	Log files and directories
`mail`	User mailbox files
`opt`	Variable data for /opt
`run`	Files relevant to running processes
`spool`	Application spool data
`state`	Variable state information
`tmp`	Temporary files preserved between system reboots
`yp`	Network Information Service (NIS) database files

`/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/mail`, `/var/cache/man`, `/var/cache/fonts`, 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) must 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`.

Applications should generally not add directories to the top level of `/var`. Such directories should only be added if they have some system-wide implication, and in consultation with the FHS mailing list.

The `cache`, `lock`, `log`, `run`, `spool`, `state`, and `tmp` directories must be included and used in all distributions; the `account`, `crash`, `cron`, `games`, `mail`, and `yp` directories must be included and used if the corresponding applications or features are provided in the distribution.

Several directories are ‘reserved’ in the sense that they should not be used arbitrarily by some new application, since they would conflict with historical and/or local practice. They are:

```
/var/backups
/var/local
/var/msgs
/var/preserve
```

5.1 /var/account : Process accounting logs (if supported)

This directory holds the current active process accounting log and the composite process usage data (as used by `lastcomm` and `sa`). As of the date of this release of the standard, process accounting was not supported under Linux.

5.2 /var/cache : Application cache data

/var/cache — Cache directories

`fonts`	Locally-generated fonts
`man`	Locally-formatted manual pages
`www`	WWW proxy or cache data
`<package>`	Package specific cache data

`/var/cache` is intended for cached data from applications. Such data is locally generated as a result of time consuming I/O or calculation. The application must be able to regenerate or restore the data. Unlike `/var/spool`, the cached files can be deleted without data loss. The data should remain valid between invocations of the application and rebooting the system.

Files located under /var/cache may be expired in an application specific manner, by the system administrator, or both. The application should always be able to recover from manual deletion of files (generally because of a disk space shortage). No other requirements are made on the data format of the cache directories.

BEGIN RATIONALE

The existance of a separate directory for cached data allows system administrators to set different disk and backup policies from other directories in `/var`.

END RATIONALE

5.2.1 /var/cache/fonts : Locally-generated fonts

The directory `/var/cache/fonts` should be used to store any dynamically-created fonts. In particular, `/var/cache/fonts/pk` 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/cache/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/cache/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/cache/fonts` 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/cache/fonts`.

---

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 free UNIX community.

Other dynamically created fonts may also be placed in this tree, under appropriately-named subdirectories of /var/cache/fonts.

5.2.2 /var/cache/man : 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/cache/man and may write formatted man pages into the cat[1-9] directories in /usr/share/man directly. We recommend that most sites use one of the following options instead:

- Preformat all manual pages alongside the unformatted versions.
- Allow no caching of formatted man pages, and require formatting to be done each time a man page is brought up.
- Allow local caching of formatted man pages in /var/cache/man.

The structure of /var/cache/man 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/<path>/man/<locale>/man[1-9], the cached formatted version should go in /var/cache/man/<path>/<locale>/cat[1-9]. (Note that the <locale> component may be missing.) For a manual page that normally appears outside /usr, in /<path>/man/<locale>/man[1-9], the cached formatted version should go in /var/cache/man/<path>/<locale>/cat[1-9]. The <path> component is absent in the case of /usr/share/man and /var/cache/man.

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

Man pages written to /var/cache/man/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 system on read-only media (a CD-ROM, for instance), they shall be installed into /usr/<path>/cat[1-9]. /var/cache/man is reserved as a writable cache for formatted manual pages.

BEGIN RATIONALE

Release 1.2 of the standard specified /var/catman for this hierarchy. The path has been changed to /var/cache to better reflect the dynamic nature of the formatted man pages. The directory name has been changed to man to allow for enhancing the hierarchy to include post-processed formats other than "cat", such as PostScript, HTML, or DVI.

END RATIONALE

5.3 /var/crash : System crash dumps (if supported)

This directory holds system crash dumps. As of the date of this release of the standard, system crash dumps were not supported under Linux.

5.4 /var/cron : cron and at jobs

This directory contains the variable data for the cron and at processes.

BEGIN RATIONALE

Previous Linux releases, as well as most commercial vendors, use `/var/spool` for `cron`. Given the definition of `/var/spool`, it was judged reasonable to move `cron` elsewhere in the `/var` hierarchy. Precedent from BSD UNIX suggested using `/var/cron` rather than `/var/state/cron`. Combining `cron` and `at` in this hierarchy assumes that a free `cron` that includes `at` functionality will be implemented in the future.

END RATIONALE

5.5 /var/games : Variable game data

Any variable data relating to games, such as those in `/usr/games`, should be placed here. `/var/games` should hold the variable data previously found in `/usr`; static data, such as help text, level descriptions, and so on, should remain elsewhere, such as `/usr/lib/games`.

BEGIN RATIONALE

`/var/games` has been given a hierarchy of its own, rather than leaving it merged in with the old `/var/lib` as in release 1.2. The separation allows local control of backup strategies, permissions, and disk usage, as well as allowing inter-host sharing and reducing clutter in `/var/state`. Additionally, `/var/games` is the path traditionally used by BSD.

END RATIONALE

5.6 /var/lock : Lock files

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

Device lock files, such as the serial device lock files that were originally found in either `/usr/spool/locks` or `/usr/spool/uucp`, must now be stored in `/var/lock`. The naming convention which must 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 device lock files must 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).

5.7 /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

5.8 /var/opt : Variable data for /opt

Variable data should be installed in `/var/opt/<package>`, where `<package>` is the name of the subtree in `/opt` where the static data from an add-on software package is stored, except where superseded by another file in `/etc`. No structure is imposed on the internal arrangement of `/var/opt/<package>`.

BEGIN RATIONALE

Refer to the rationale for `/opt`.

END RATIONALE

5.9 /var/run : Run-time variable files

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

Process identifier (PID) files, which were originally placed in `/etc`, should be 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.

5.10 /var/spool : Application spool data

/var/spool — Spool directories

`lpd`	Printer spool directory
`mqueue`	Outgoing mail queue
`news`	News spool directory
`rwho`	Rwhod files
`smail`	Spool directories for smail
`uucp`	Spool directory for UUCP

`/var/spool` contains data which is awaiting some kind of later processing. Data in `/var/spool` represents work to be done in the future (by a program, user, or administrator); often data is deleted after it

has been processed.

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

5.10.1 /var/spool/lpd : Line-printer daemon print queues

/var/spool/lpd — Printer spool directory

`<printer>`	Spools for a specific printer (optional)

The lock file for `lpd`, `lpd.lock`, should be placed in `/var/spool/lpd`. It is suggested that the lock file for each printer be placed in the spool directory for that specific printer and named `lock`.

5.10.2 /var/spool/rwho : Rwhod files

This directory holds the rwhod information for other systems on the local net.

BEGIN RATIONALE

Previous BSD releases used `/var/rwho` for this data; given its close fit to the definition of 'spooled' data, and its historical precedent in `/var/spool` on other systems, a move was deemed appropriate.

END RATIONALE

5.11 /var/state : Variable state information

/var/state — Variable state information

`<editor>`	Saved files after crash and other editor state
`misc`	Miscellaneous state data
`xdm`	X display manager variable data
`<pkgtool>`	Packaging support files
`<package>`	State data for packages and subsystems

This hierarchy holds state information pertaining to an application or the system. State information is data that programs modify while they run, and that pertains to one specific host. Users should never need to modify files in `/var/state` to configure a package's operation.

State information is generally used to preserve the condition of an application (or a group of inter-related applications) between invocations and between different instances of the same application. State information should generally remain valid after a reboot, should not be logging output, and should not be spooled data.

An application (or a group of inter-related applications) should use a subdirectory of `/var/state` for its data. There is one required subdirectory, `/var/state/misc`, which is intended for state files that don't

need a subdirectory; the other subdirectories should only be present if the application in question is included in the distribution.

/var/state/<name> is the location that should be used for all distribution packaging support. Different distributions may utilize different names, of course.

BEGIN RATIONALE

Previous releases of the standard used the name /var/lib. However, /usr/lib is increasingly used solely for object files or archives of them; this is true of the current BSD UNIX variants as well as current GNU packages. Accordingly, the name /var/lib seemed inappropriate.

BSD used the name /var/db for a similar directory. This name seemed overly constricting, as it implied a directory structure intended primarily for database (.db) files.

END RATIONALE

5.11.1 /var/state/<editor> : Saved files after crash and other editor state

These directories contain saved files generated by any unexpected termination of an editor (e.g., elvis, jove, nvi).

Other editors may not require a directory for crash-recovery files, but may require a well-defined place to store other information while the editor is running. This information should be stored in a subdirectory under /var/state (for example, GNU Emacs would place lock files in /var/state/emacs/lock).

Future editors may require additional state information beyond crash-recovery files and lock files -- this information should also be placed under /var/state/<editor>.

BEGIN RATIONALE

Previous Linux releases, as well as all commercial vendors, use /var/preserve for vi or its clones. However, each editor uses its own format for these crash-recovery files, so a separate directory is needed for each editor.

Editor-specific lock files are usually quite different from the device or resource lock files that are stored in /var/lock and, hence, are stored under /var/state.

END RATIONALE

5.11.2 /var/state/misc : Miscellaneous variable data

This directory contains variable data not placed in a subdirectory in /var/state. An attempt should be made to use relatively unique names in this directory to avoid namespace conflicts.

Note that files previously stored in /var/db in previous BSD releases are moved here. These include locate.database and mountdtab, and the kernel symbol database(s).

5.12 /var/tmp : Temporary files preserved between system reboots

The /var/tmp directory is made available for programs that require temporary files or directories that are preserved between system reboots. Therefore, data stored in /var/tmp is more persistent than data in /tmp.

Files and directories located in `/var/tmp` must not be deleted when the system is booted. Although data stored in `/var/tmp` is typically deleted in a site-specific manner, it is recommended that deletions occur at a less frequent interval than `/tmp`.

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

Variable data for the Network Information Service (NIS), formerly known as the Sun Yellow Pages (YP), shall be placed in this directory.

BEGIN RATIONALE

`/var/yp` is the standard directory for NIS (YP) data and is almost exclusively used in NIS documentation and systems.

NIS should not be confused with Sun NIS+, which uses a different directory, `/var/nis`.

END RATIONALE

6. Operating System Specific Annex

This section is for additional requirements and recommendations that only apply to a specific operating system. The material in this section should never conflict with the base standard.

6.1 Linux

This is the annex for the Linux operating system.

6.1.1 /

On Linux systems, if the kernel is located in `/`, we recommend using the names `vmlinux` or `vmlinuz`, which have been used in recent Linux kernel source packages.

6.1.2 /dev

All devices and special files in `/dev` should adhere to the *Linux Allocated Devices* document, which is available with the Linux kernel source. It is maintained by H. Peter Anvin <hpa@zytor.com>.

Symbolic links in `/dev` should not be distributed with Linux systems except as provided in the *Linux Allocated Devices* document.

BEGIN RATIONALE

The requirement not to make symlinks promiscuously is made 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.

END RATIONALE

6.1.3 /sbin

Linux systems place these additional files into `/sbin`.

- Second extended filesystem commands (optional):

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

- Boot-loader map installer:

  `{ lilo }`

Optional files for /sbin:

- Static binaries:

  `{ ldconfig, sln, ssync }`

  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 linked 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.

- Miscellaneous:

`{ ctrlaltdel, kbdrate }`

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.

6.1.4 /usr/include

These symbolic links are required if a C or C++ compiler is installed.

```
/usr/include/asm -> /usr/src/linux/include/asm-<arch>
/usr/include/linux -> /usr/src/linux/include/linux
```

6.1.5 /usr/src

The only source code that should be placed in a specific location is the Linux kernel source code. It is located in `/usr/src/linux`.

If a C or C++ compiler is installed, but the complete Linux kernel source code is not installed, then the include files from the kernel source code shall be located in these directories:

```
/usr/src/linux/include/asm-<arch>
/usr/src/linux/include/linux
```

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

BEGIN RATIONALE

It is important that the kernel include files be located in `/usr/src/linux` and not in `/usr/include` so there are no problems when system administrators upgrade their kernel version for the first time.

END RATIONALE

The FHS mailing list

The FHS mailing list is located at <fhs-discuss@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 fhs-discuss`".)

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 FHS editor or a listed contributor.

Acknowledgments

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

We (Daniel Quinlan and the FHS group) express our gratitude for the support of 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; and to the intrepid BSD folk who answered our call to help develop a wider standard addressing the needs of both communities.

We especially wish to note the bravery and perseverance of those Linux developers who started following drafts of the ancestral FSSTND before it was even completed. Without the demonstration they provided that the FSSTND was worthwhile, the FHS could never have evolved.

Original contributors

Drew Eckhardt <drew@colorado.edu>
Ian Jackson <ijackson@cus.cam.ac.uk>
Ian McCloghrie <ian@ucsd.edu>
Daniel Quinlan <quinlan@pathname.com>
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>
Keith Bostic <bostic@cs.berkeley.edu>
Rik Faith <faith@cs.unc.edu>
Stephen Harris <sweh@spuddy.mew.co.uk>
Fred N. van Kempen <waltje@infomagic.com>
John A. Martin <jmartin@acm.org>
Chris Metcalf <metcalf@lcs.mit.edu>
Ian Murdock <imurdock@debian.org>
David C. Niemi <niemidc@clark.net>
Eric S. Raymond <esr@thyrsus.com>

CONTENTS

1. Introduction 1
   - 1.1 Status of the Standard 1
   - 1.2 Organization of the Standard 1
   - 1.3 Conventions 1
   - 1.4 Background of the FHS 2
   - 1.5 Scope 2
   - 1.6 Objectives 3
   - 1.7 Conformance with this Document 3

2. The Filesystem 5

3. The Root Directory 7
   - 3.1 /bin : Essential user command binaries (for use by all users) 8
   - 3.2 /boot : Static files of the boot loader 9
   - 3.3 /dev : Device files 10
   - 3.4 /etc : Host-specific system configuration 10
   - 3.5 /home : User home directories (optional) 11
   - 3.6 /lib : Essential shared libraries and kernel modules 11
   - 3.7 /mnt : Mount point for temporarily mounted filesystems 12
   - 3.8 /opt : Add-on application software packages 12
   - 3.9 /proc : Kernel and process information virtual filesystem 13
   - 3.10 /root : Home directory for the root user (optional) 13
   - 3.11 /sbin : System binaries (binaries once kept in /etc) 13
   - 3.12 /tmp : Temporary files 14

4. The /usr Hierarchy 16
   - 4.1 /usr/X11R6 : X Window System, Version 11 Release 6 17
   - 4.2 /usr/X386 : X Window System, Version 11 Release 5, on x86 platforms 17
   - 4.3 /usr/bin : Most user commands 17
   - 4.4 /usr/include : Directory for standard include files. 18
   - 4.5 /usr/lib : Libraries for programming 18
   - 4.6 /usr/libexec : Internal binaries 19
   - 4.7 /usr/local : Local hierarchy 19
   - 4.8 /usr/sbin : Non-essential standard system binaries 20
   - 4.9 /usr/share : Architecture-independent data 20
   - 4.10 /usr/src : Source code 25

5. The /var Hierarchy 26
   - 5.1 /var/account : Process accounting logs (if supported) 27
   - 5.2 /var/cache : Application cache data 27
   - 5.3 /var/crash : System crash dumps (if supported) 28
   - 5.4 /var/cron : cron and at jobs 29
   - 5.5 /var/games : Variable game data 29
   - 5.6 /var/lock : Lock files 29
   - 5.7 /var/log : Log files and directories 29
   - 5.8 /var/opt : Variable data for /opt 30
   - 5.9 /var/run : Run-time variable files 30
   - 5.10 /var/spool : Application spool data 30
   - 5.11

/var/state : Variable state information 31
5.12 /var/tmp : Temporary files preserved between system reboots 32
5.13 /var/yp : Network Information Service (NIS) database files 33
6. Operating System Specific Annex 34
6.1 Linux 34