Filesystem Hierarchy Standard

Filesystem Hierarchy Standard Group

Edited by

Rusty Russell

Daniel Quinlan

Christopher Yeoh

Filesystem Hierarchy Standard
by Filesystem Hierarchy Standard Group
Edited by Rusty Russell, Daniel Quinlan, and Christopher Yeoh

Published January 28 2004

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.

Table of Contents

1. Introduction **1**
1.1. Purpose 1
1.2. Conventions 1
2. The Filesystem **2**
3. The Root Filesystem **3**
3.1. Purpose 3
3.2. Requirements 4
3.3. Specific Options 4
3.4. /bin : Essential user command binaries (for use by all users) 4
3.4.1. Purpose 4
3.4.2. Requirements 5
3.4.3. Specific Options 6
3.5. /boot : Static files of the boot loader 6
3.5.1. Purpose 6
3.5.2. Specific Options 7
3.6. /dev : Device files 7
3.6.1. Purpose 7
3.6.2. Specific Options 7
3.7. /etc : Host-specific system configuration 7
3.7.1. Purpose 7
3.7.2. Requirements 7
3.7.3. Specific Options 7
3.7.4. /etc/opt : Configuration files for /opt 8
3.7.4.1. Purpose 9
3.7.4.2. Requirements 9
3.7.5. /etc/X11 : Configuration for the X Window System (optional) 9
3.7.5.1. Purpose 9
3.7.5.2. Specific Options 9
3.7.6. /etc/sgml : Configuration files for SGML (optional) 9
3.7.6.1. Purpose 10
3.7.7. /etc/xml : Configuration files for XML (optional) 10
3.7.7.1. Purpose 10
3.8. /home : User home directories (optional) 10
3.8.1. Purpose 10
3.8.2. Requirements 10
3.9. /lib : Essential shared libraries and kernel modules 10
3.9.1. Purpose 10
3.9.2. Requirements 10
3.9.3. Specific Options 11
3.10. /lib<qual> : Alternate format essential shared libraries (optional) 11
3.10.1. Purpose 11
3.10.2. Requirements 11
3.11. /media : Mount point for removeable media 11
3.11.1. Purpose 11
3.11.2. Specific Options 12
3.12. /mnt : Mount point for a temporarily mounted filesystem 12
3.12.1. Purpose 12
3.13. /opt : Add-on application software packages 12
3.13.1. Purpose 12

3.13.2. Requirements 12
3.14. /root : Home directory for the root user (optional)........ 13
3.14.1. Purpose 14
3.15. /sbin : System binaries........ 14
3.15.1. Purpose 14
3.15.2. Requirements 14
3.15.3. Specific Options........ 14
3.16. /srv : Data for services provided by this system........ 15
3.16.1. Purpose 15
3.17. /tmp : Temporary files 15
3.17.1. Purpose 15
4. The /usr Hierarchy........ 18
4.1. Purpose........ 18
4.2. Requirements........ 18
4.3. Specific Options 18
4.4. /usr/X11R6 : X Window System, Version 11 Release 6 (optional)........ 18
4.4.1. Purpose 19
4.4.2. Specific Options........ 19
4.5. /usr/bin : Most user commands 19
4.5.1. Purpose 19
4.5.2. Specific Options........ 19
4.6. /usr/include : Directory for standard include files. 20
4.6.1. Purpose 20
4.6.2. Specific Options........ 20
4.7. /usr/lib : Libraries for programming and packages 20
4.7.1. Purpose 20
4.7.2. Specific Options........ 20
4.8. /usr/lib<qual> : Alternate format libraries (optional)........ 21
4.8.1. Purpose 21
4.8.2. /usr/local : Local hierarchy 21
4.8.2.1. Purpose........ 21
4.8.2.2. Requirements 21
4.8.2.3. Specific Options 21
4.9. /usr/local/share 22
4.10. /usr/sbin : Non-essential standard system binaries........ 22
4.10.1. Purpose 22
4.11. /usr/share : Architecture-independent data........ 22
4.11.1. Purpose 22
4.11.2. Requirements 22
4.11.3. Specific Options........ 23
4.11.4. /usr/share/dict : Word lists (optional)........ 23
4.11.4.1. Purpose........ 23
4.11.4.2. Specific Options 23
4.11.5. /usr/share/man : Manual pages 24
4.11.5.1. Purpose........ 24
4.11.5.2. Specific Options 25
4.11.6. /usr/share/misc : Miscellaneous architecture-independent data 26
4.11.6.1. Specific Options 26
4.11.7. /usr/share/sgml : SGML data (optional) 27
4.11.7.1. Purpose........ 27
4.11.7.2. Specific Options 27

4.11.8. /usr/share/xml : XML data (optional) 27
4.11.8.1. Purpose 27
4.11.8.2. Specific Options 27
4.12. /usr/src : Source code (optional) 28
4.12.1. Purpose 28
5. The /var Hierarchy 30
5.1. Purpose 30
5.2. Requirements 30
5.3. Specific Options 30
5.4. /var/account : Process accounting logs (optional) 31
5.4.1. Purpose 31
5.5. /var/cache : Application cache data 31
5.5.1. Purpose 31
5.5.2. Specific Options 31
5.5.3. /var/cache/fonts : Locally-generated fonts (optional) 32
5.5.3.1. Purpose 32
5.5.3.2. Specific Options 32
5.5.4. /var/cache/man : Locally-formatted manual pages (optional) 32
5.5.4.1. Purpose 32
5.6. /var/crash : System crash dumps (optional) 33
5.6.1. Purpose 33
5.7. /var/games : Variable game data (optional) 33
5.7.1. Purpose 33
5.8. /var/lib : Variable state information 33
5.8.1. Purpose 33
5.8.2. Requirements 34
5.8.3. Specific Options 34
5.8.4. /var/lib/<editor> : Editor backup files and state (optional) 34
5.8.4.1. Purpose 34
5.8.5. /var/lib/hwclock : State directory for hwclock (optional) 35
5.8.5.1. Purpose 35
5.8.6. /var/lib/misc : Miscellaneous variable data 35
5.8.6.1. Purpose 35
5.9. /var/lock : Lock files 35
5.9.1. Purpose 35
5.10. /var/log : Log files and directories 35
5.10.1. Purpose 36
5.10.2. Specific Options 36
5.11. /var/mail : User mailbox files (optional) 36
5.11.1. Purpose 36
5.12. /var/opt : Variable data for /opt 36
5.12.1. Purpose 36
5.13. /var/run : Run-time variable data 37
5.13.1. Purpose 37
5.13.2. Requirements 37
5.14. /var/spool : Application spool data 37
5.14.1. Purpose 37
5.14.2. Specific Options 37
5.14.3. /var/spool/lpd : Line-printer daemon print queues (optional) 38
5.14.3.1. Purpose 38
5.14.3.2. Specific Options 38

5.14.4. /var/spool/rwho : Rwhod files (optional) ... 38
5.14.4.1. Purpose ... 38
5.15. /var/tmp : Temporary files preserved between system reboots ... 38
5.15.1. Purpose ... 38
5.16. /var/yp : Network Information Service (NIS) database files (optional) ... 39
5.16.1. Purpose ... 39
6. Operating System Specific Annex ... 40
6.1. Linux ... 40
6.1.1. / : Root directory ... 40
6.1.2. /bin : Essential user command binaries (for use by all users) ... 40
6.1.3. /dev : Devices and special files ... 40
6.1.4. /etc : Host-specific system configuration ... 40
6.1.5. /lib64 and /lib32 : 64/32-bit libraries (architecture dependent) ... 41
6.1.6. /proc : Kernel and process information virtual filesystem ... 41
6.1.7. /sbin : Essential system binaries ... 41
6.1.8. /usr/include : Header files included by C programs ... 42
6.1.9. /usr/src : Source code ... 42
6.1.10. /var/spool/cron : cron and at jobs ... 43
7. Appendix ... 44
7.1. The FHS mailing list ... 44
7.2. Background of the FHS ... 44
7.3. General Guidelines ... 44
7.4. Scope ... 44
7.5. Acknowledgments ... 45
7.6. Contributors ... 45

Chapter 1. Introduction

1.1. Purpose

This standard enables:

- Software to predict the location of installed files and directories, and
- Users to predict the location of installed files and directories.

We do this by:

- Specifying guiding principles for each area of the filesystem,
- Specifying the minimum files and directories required,
- Enumerating exceptions to the principles, and
- Enumerating specific cases where there has been historical conflict.

The FHS document is used by:

- Independent software suppliers to create applications which are FHS compliant, and work with distributions which are FHS complaint,
- OS creators to provide systems which are FHS compliant, and
- Users to understand and maintain the FHS compliance of a system.

The FHS document has a limited scope:

- Local placement of local files is a local issue, so FHS does not attempt to usurp system administrators.
- FHS addresses issues where file placements need to be coordinated between multiple parties such as local sites, distributions, applications, documentation, etc.

1.2. Conventions

We recommend that you read a typeset version of this document rather than the plain text version. In the typeset version, the names of files and directories are displayed in a constant-width font.

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 filename is allowed to occur either with or without an extension, it might be represented by *<filename>[.<extension>]*.

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

The sections of the text marked as *Rationale* are explanatory and are non-normative.

Chapter 2. The Filesystem

This standard assumes that the operating system underlying an FHS-compliant file system supports the same basic security features found in most UNIX filesystems.

It is possible to define two independent distinctions among files: shareable vs. unshareable and variable vs. static. In general, files that differ in either of these respects should be located in different directories. This makes it easy to store files with different usage characteristics on different filesystems.

"Shareable" files are those that can be stored on one host and used on others. "Unshareable" files are those that are not shareable. For example, the files in user home directories are shareable whereas device lock files are not.

"Static" files include binaries, libraries, documentation files and other files that do not change without system administrator intervention. "Variable" files are files that are not static.

Rationale

Shareable files can be stored on one host and used on several others. Typically, however, not all files in the filesystem hierarchy are shareable and so each system has local storage containing at least its unshareable files. It is convenient if all the files a system requires that are stored on a foreign host can be made available by mounting one or a few directories from the foreign host.

Static and variable files should be segregated because static files, unlike variable files, can be stored on read-only media and do not need to be backed up on the same schedule as variable files.

Historical UNIX-like filesystem hierarchies contained both static and variable files under both `/usr` and `/etc`. In order to realize the advantages mentioned above, the `/var` hierarchy was created and all variable files were transferred from `/usr` to `/var`. Consequently `/usr` can now be mounted read-only (if it is a separate filesystem). Variable files have been transferred from `/etc` to `/var` over a longer period as technology has permitted.

Here is an example of a FHS-compliant system. (Other FHS-compliant layouts are possible.)

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

Chapter 3. The Root Filesystem

3.1. Purpose

The contents of the root filesystem must 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 partitions or filesystems.
- To enable recovery and/or repair of a system, those utilities needed by an experienced maintainer to diagnose and reconstruct a damaged system must be present on the root filesystem.
- To restore a system, those utilities needed to restore from system backups (on floppy, tape, etc.) must be present on the root filesystem.

Rationale

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 contains many system-specific configuration files. Possible examples include a kernel that is specific to the system, a specific hostname, etc. This means that the root filesystem isn't always shareable between networked systems. Keeping it small on servers in networked systems minimizes the amount of lost space for areas of 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.

Applications must 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.

Rationale

There are several reasons why creating a new subdirectory of the root filesystem 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.

Distributions should not create new directories in the root hierarchy without extremely careful consideration of the consequences including for application portability.

3.2. Requirements

The following directories, or symbolic links to directories, are required in `/`.

Directory	Description
`bin`	Essential command binaries
`boot`	Static files of the boot loader
`dev`	Device files
`etc`	Host-specific system configuration
`lib`	Essential shared libraries and kernel modules
`media`	Mount point for removeable media
`mnt`	Mount point for mounting a filesystem temporarily
`opt`	Add-on application software packages
`sbin`	Essential system binaries
`srv`	Data for services provided by this system
`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.

3.3. Specific Options

The following directories, or symbolic links to directories, must be in `/`, if the corresponding subsystem is installed:

Directory	Description
`home`	User home directories (optional)
`lib<qual>`	Alternate format essential shared libraries (optional)
`root`	Home directory for the root user (optional)

Each directory listed above is specified in detail in separate subsections below.

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

3.4.1. Purpose

`/bin` contains commands that may be used by both the system administrator and by users, but which are required when no other filesystems are mounted (e.g. in single user mode). It may also contain commands which are used indirectly by scripts. [1]

3.4.2. Requirements

There must be no subdirectories in `/bin`.

The following commands, or symbolic links to commands, are required in `/bin`.

Command	Description
cat	Utility to concatenate files to standard output
chgrp	Utility to change file group ownership
chmod	Utility to change file access permissions
chown	Utility to change file owner and group
cp	Utility to copy files and directories
date	Utility to print or set the system data and time
dd	Utility to convert and copy a file
df	Utility to report filesystem disk space usage
dmesg	Utility to print or control the kernel message buffer
echo	Utility to display a line of text
false	Utility to do nothing, unsuccessfully
hostname	Utility to show or set the system's host name
kill	Utility to send signals to processes
ln	Utility to make links between files
login	Utility to begin a session on the system
ls	Utility to list directory contents
mkdir	Utility to make directories
mknod	Utility to make block or character special files
more	Utility to page through text
mount	Utility to mount a filesystem
mv	Utility to move/rename files
ps	Utility to report process status
pwd	Utility to print name of current working directory
rm	Utility to remove files or directories
rmdir	Utility to remove empty directories
sed	The 'sed' stream editor
sh	The Bourne command shell
stty	Utility to change and print terminal line settings
su	Utility to change user ID
sync	Utility to flush filesystem buffers
true	Utility to do nothing, successfully
umount	Utility to unmount file systems
uname	Utility to print system information

If **/bin/sh** is not a true Bourne shell, it must be a hard or symbolic link to the real shell command.

The **[** and **test** commands must be placed together in either `/bin` or `/usr/bin`.

Rationale

For example bash behaves differently when called as **sh** or **bash**. The use of a symbolic link also allows users to easily see that /**bin**/**sh** is not a true Bourne shell.

The requirement for the **[** and **test** commands to be included as binaries (even if implemented internally by the shell) is shared with the POSIX.2 standard.

3.4.3. Specific Options

The following programs, or symbolic links to programs, must be in `/bin` if the corresponding subsystem is installed:

Command	Description
csh	The C shell (optional)
ed	The ‘ed’ editor (optional)
tar	The tar archiving utility (optional)
cpio	The cpio archiving utility (optional)
gzip	The GNU compression utility (optional)
gunzip	The GNU uncompression utility (optional)
zcat	The GNU uncompression utility (optional)
netstat	The network statistics utility (optional)
ping	The ICMP network test utility (optional)

If the **gunzip** and **zcat** programs exist, they must be symbolic or hard links to gzip. **/bin/csh** may be a symbolic link to **/bin/tcsh** or **/usr/bin/tcsh**.

Rationale

The tar, gzip and cpio commands have been added to make restoration of a system possible (provided that `/` is intact).

Conversely, if no restoration from the root partition is ever expected, then these binaries might be omitted (e.g., a ROM chip root, mounting `/usr` through NFS). If restoration of a system is planned through the network, then **ftp** or **tftp** (along with everything necessary to get an ftp connection) must be available on the root partition.

3.5. /boot : Static files of the boot loader

3.5.1. Purpose

This directory contains everything required for the boot process except configuration files not needed at boot time and the map installer. Thus /boot stores data that is used before the kernel begins executing user-mode programs. This may include saved master boot sectors and sector map files. [2]

3.5.2. Specific Options

The operating system kernel must be located in either `/` or `/boot`. [3]

3.6. /dev : Device files

3.6.1. Purpose

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

3.6.2. Specific Options

If it is possible that devices in `/dev` will need to be manually created, `/dev` must 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` must have provisions for creating any device that may be found on the system, not just those that a particular implementation installs.

3.7. /etc : Host-specific system configuration

3.7.1. Purpose

The `/etc` hierarchy contains configuration files. A "configuration file" is a local file used to control the operation of a program; it must be static and cannot be an executable binary. [4]

3.7.2. Requirements

No binaries may be located under `/etc`. [5]

The following directories, or symbolic links to directories are required in `/etc`:

Directory	Description
opt	Configuration for /opt
X11	Configuration for the X Window system (optional)
sgml	Configuration for SGML (optional)
xml	Configuration for XML (optional)

3.7.3. Specific Options

The following directories, or symbolic links to directories must be in `/etc`, if the corresponding subsystem is installed:

Directory	Description
opt	Configuration for /opt

The following files, or symbolic links to files, must be in `/etc` if the corresponding subsystem is installed: [6]

File	Description
`csh.login`	Systemwide initialization file for C shell logins (optional)
`exports`	NFS filesystem access control list (optional)
`fstab`	Static information about filesystems (optional)
`ftpusers`	FTP daemon user access control list (optional)
`gateways`	File which lists gateways for routed (optional)
`gettydefs`	Speed and terminal settings used by getty (optional)
`group`	User group file (optional)
`host.conf`	Resolver configuration file (optional)
`hosts`	Static information about host names (optional)
`hosts.allow`	Host access file for TCP wrappers (optional)
`hosts.deny`	Host access file for TCP wrappers (optional)
`hosts.equiv`	List of trusted hosts for rlogin, rsh, rcp (optional)
`hosts.lpd`	List of trusted hosts for lpd (optional)
`inetd.conf`	Configuration file for inetd (optional)
`inittab`	Configuration file for init (optional)
`issue`	Pre-login message and identification file (optional)
`ld.so.conf`	List of extra directories to search for shared libraries (optional)
`motd`	Post-login message of the day file (optional)
`mtab`	Dynamic information about filesystems (optional)
`mtools.conf`	Configuration file for mtools (optional)
`networks`	Static information about network names (optional)
`passwd`	The password file (optional)
`printcap`	The lpd printer capability database (optional)
`profile`	Systemwide initialization file for sh shell logins (optional)
`protocols`	IP protocol listing (optional)
`resolv.conf`	Resolver configuration file (optional)
`rpc`	RPC protocol listing (optional)
`securetty`	TTY access control for root login (optional)
`services`	Port names for network services (optional)
`shells`	Pathnames of valid login shells (optional)
`syslog.conf`	Configuration file for syslogd (optional)

`mtab` does not fit the static nature of `/etc`: it is excepted for historical reasons. [7]

3.7.4. /etc/opt : Configuration files for /opt

3.7.4.1. Purpose

Host-specific configuration files for add-on application software packages must be installed within the directory `/etc/opt/<subdir>`, where `<subdir>` is the name of the subtree in `/opt` where the static data from that package is stored.

3.7.4.2. Requirements

No structure is imposed on the internal arrangement of `/etc/opt/<subdir>`.

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/<subdir>`.

Rationale

Refer to the rationale for `/opt`.

3.7.5. /etc/X11 : Configuration for the X Window System (optional)

3.7.5.1. Purpose

/etc/X11 is the location for all X11 host-specific configuration. This directory is necessary to allow local control if */usr* is mounted read only.

3.7.5.2. Specific Options

The following files, or symbolic links to files, must be in `/etc/X11` if the corresponding subsystem is installed:

File	Description
`Xconfig`	The configuration file for early versions of XFree86 (optional)
`XF86Config`	The configuration file for XFree86 versions 3 and 4 (optional)
`Xmodmap`	Global X11 keyboard modification file (optional)

Subdirectories of `/etc/X11` may include those for `xdm` and for any other programs (some window managers, for example) that need them. [8] We recommend that window managers with only one configuration file which is a default `.*wmrc` file must name it `system.*wmrc` (unless there is a widely-accepted alternative name) and not use a subdirectory. Any window manager subdirectories must be identically named to the actual window manager binary.

3.7.6. /etc/sgml : Configuration files for SGML (optional)

3.7.6.1. Purpose

Generic configuration files defining high-level parameters of the SGML systems are installed here. Files with names `*.conf` indicate generic configuration files. File with names `*.cat` are the DTD-specific centralized catalogs, containing references to all other catalogs needed to use the given DTD. The super catalog file `catalog` references all the centralized catalogs.

3.7.7. /etc/xml : Configuration files for XML (optional)

3.7.7.1. Purpose

Generic configuration files defining high-level parameters of the XML systems are installed here. Files with names `*.conf` indicate generic configuration files. The super catalog file `catalog` references all the centralized catalogs.

3.8. /home : User home directories (optional)

3.8.1. Purpose

`/home` is a fairly standard concept, but it is clearly a site-specific filesystem. [9] The setup will differ from host to host. Therefore, no program should rely on this location. [10]

3.8.2. Requirements

User specific configuration files for applications are stored in the user's home directory in a file that starts with the '.' character (a "dot file"). If an application needs to create more than one dot file then they should be placed in a subdirectory with a name starting with a '.' character, (a "dot directory"). In this case the configuration files should not start with the '.' character. [11]

3.9. /lib : Essential shared libraries and kernel modules

3.9.1. Purpose

The `/lib` directory contains those shared library images needed to boot the system and run the commands in the root filesystem, ie. by binaries in `/bin` and `/sbin`. [12]

3.9.2. Requirements

At least one of each of the following filename patterns are required (they may be files, or symbolic links):

File	Description
libc.so.*	The dynamically-linked C library (optional)
ld*	The execution time linker/loader (optional)

If a C preprocessor is installed, */lib/cpp* must be a reference to it, for historical reasons. [13]

3.9.3. Specific Options

The following directories, or symbolic links to directories, must be in /lib, if the corresponding subsystem is installed:

Directory	Description
modules	Loadable kernel modules (optional)

3.10. /lib<qual> : Alternate format essential shared libraries (optional)

3.10.1. Purpose

There may be one or more variants of the /lib directory on systems which support more than one binary format requiring separate libraries. [14]

3.10.2. Requirements

If one or more of these directories exist, the requirements for their contents are the same as the normal /lib directory, except that /lib<qual>/cpp is not required. [15]

3.11. /media : Mount point for removeable media

3.11.1. Purpose

This directory contains subdirectories which are used as mount points for removeable media such as floppy disks, cdroms and zip disks.

Rationale

> Historically there have been a number of other different places used to mount removeable media such as `/cdrom`, `/mnt` or `/mnt/cdrom`. Placing the mount points for all removeable media directly in the root directory would potentially result in a large number of extra directories in `/`. Although the use of subdirectories in `/mnt` as a mount point has recently been common, it conflicts with a much older tradition of using `/mnt` directly as a temporary mount point.

3.11.2. Specific Options

The following directories, or symbolic links to directories, must be in `/media`, if the corresponding subsystem is installed:

Directory	Description
`floppy`	Floppy drive (optional)
`cdrom`	CD-ROM drive (optional)
`cdrecorder`	CD writer (optional)
`zip`	Zip drive (optional)

On systems where more than one device exists for mounting a certain type of media, mount directories can be created by appending a digit to the name of those available above starting with '0', but the unqualified name must also exist. [16]

3.12. /mnt : Mount point for a temporarily mounted filesystem

3.12.1. Purpose

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

This directory must not be used by installation programs: a suitable temporary directory not in use by the system must be used instead.

3.13. /opt : Add-on application software packages

3.13.1. Purpose

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

A package to be installed in `/opt` must locate its static files in a separate `/opt/<package>` or `/opt/<provider>` directory tree, where `<package>` is a name that describes the software package and `<provider>` is the provider's LANANA registered name.

3.13.2. Requirements

Directory	Description
<package>	Static package objects
<provider>	LANANA registered provider name

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 must function normally in the absence of these reserved directories.

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

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

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

No other package files may 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.

Distributions may install software in /opt, but must not modify or delete software installed by the local system administrator without the assent of the local system administrator.

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 support a package on a system must 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.

The minor restrictions on distributions using /opt are necessary because conflicts are possible between distribution-installed and locally-installed software, especially in the case of fixed pathnames found in some binary software.

The structure of the directories below /opt/<provider> is left up to the packager of the software, though it is recommended that packages are installed in /opt/<provider>/<package> and follow a similar structure to the guidelines for /opt/package. A valid reason for diverging from this structure is for support packages which may have files installed in /opt/<provider>/lib or /opt/<provider>/bin.

3.14. /root : Home directory for the root user (optional)

3.14.1. Purpose

The root account's home directory may be determined by developer or local preference, but this is the recommended default location. [17]

3.15. /sbin : System binaries

3.15.1. Purpose

Utilities used for system administration (and other root-only commands) are stored in `/sbin`, `/usr/sbin`, and `/usr/local/sbin`. `/sbin` contains binaries essential for booting, restoring, recovering, and/or repairing the system in addition to the binaries in `/bin`. [18] Programs executed after `/usr` is known to be mounted (when there are no problems) are generally placed into `/usr/sbin`. Locally-installed system administration programs should be placed into `/usr/local/sbin`. [19]

3.15.2. Requirements

The following commands, or symbolic links to commands, are required in `/sbin`.

Command	Description
shutdown	Command to bring the system down.

3.15.3. Specific Options

The following files, or symbolic links to files, must be in `/sbin` if the corresponding subsystem is installed:

Command	Description
`fastboot`	Reboot the system without checking the disks (optional)
`fasthalt`	Stop the system without checking the disks (optional)
`fdisk`	Partition table manipulator (optional)
`fsck`	File system check and repair utility (optional)
`fsck.*`	File system check and repair utility for a specific filesystem (optional)
`getty`	The getty program (optional)
`halt`	Command to stop the system (optional)
`ifconfig`	Configure a network interface (optional)
`init`	Initial process (optional)
`mkfs`	Command to build a filesystem (optional)
`mkfs.*`	Command to build a specific filesystem (optional)
`mkswap`	Command to set up a swap area (optional)

Command	Description
`reboot`	Command to reboot the system (optional)
`route`	IP routing table utility (optional)
`swapon`	Enable paging and swapping (optional)
`swapoff`	Disable paging and swapping (optional)
`update`	Daemon to periodically flush filesystem buffers (optional)

3.16. /srv : Data for services provided by this system

3.16.1. Purpose

`/srv` contains site-specific data which is served by this system.

Rationale

This main purpose of specifying this is so that users may find the location of the data files for particular service, and so that services which require a single tree for readonly data, writable data and scripts (such as cgi scripts) can be reasonably placed. Data that is only of interest to a specific user should go in that users' home directory.

The methodology used to name subdirectories of `/srv` is unspecified as there is currently no consensus on how this should be done. One method for structuring data under `/srv` is by protocol, eg. `ftp`, `rsync`, `www`, and `cvs`. On large systems it can be useful to structure `/srv` by administrative context, such as `/srv/physics/www`, `/srv/compsci/cvs`, etc. This setup will differ from host to host. Therefore, no program should rely on a specific subdirectory structure of `/srv` existing or data necessarily being stored in `/srv`. However `/srv` should always exist on FHS compliant systems and should be used as the default location for such data.

Distributions must take care not to remove locally placed files in these directories without administrator permission. [20]

3.17. /tmp : Temporary files

3.17.1. Purpose

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

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

Rationale

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

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.

FHS added this recommendation 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.

Notes

1. Command binaries that are not essential enough to place into `/bin` must be placed in `/usr/bin`, instead. Items that are required only by non-root users (the X Window System, `chsh`, etc.) are generally not essential enough to be placed into the root partition.
2. Programs necessary to arrange for the boot loader to be able to boot a file must be placed in `/sbin`. Configuration files for boot loaders must be placed in `/etc`.

   The GRUB bootloader reads its configurations file before booting, so that must be placed in `/boot`. However, it is a configuration file, so should be in `/etc`. The answer here is a symbolic link such as `/etc/grub/menu.lst` -> `/boot/menu.lst`.
3. On some i386 machines, it may be necessary for `/boot` to be located on a separate partition located completely below cylinder 1024 of the boot device due to hardware constraints.

   Certain MIPS systems require a `/boot` partition that is a mounted MS-DOS filesystem or whatever other filesystem type is accessible for the firmware. This may result in restrictions with respect to usable filenames within `/boot` (only for affected systems).
4. The setup of command scripts invoked at boot time may resemble System V, BSD or other models. Further specification in this area may be added to a future version of this standard.
5. It is recommended that files be stored in subdirectories of `/etc` rather than directly in `/etc`.
6. 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).
7. On some Linux systems, this may be a symbolic link to `/proc/mounts`, in which case this exception is not required.
8. `/etc/X11/xdm` holds the configuration files for `xdm`. These are most of the files previously found in `/usr/lib/X11/xdm`. Some local variable data for `xdm` is stored in `/var/lib/xdm`.
9. Different people prefer to place user accounts in a variety of places. 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.
10. 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.
11. It is recommended that apart from autosave and lock files programs should refrain from creating non dot files or directories in a home directory without user intervention.

12. Shared libraries that are only necessary for binaries in `/usr` (such as any X Window binaries) must not be in `/lib`. Only the shared libraries required to run binaries in `/bin` and `/sbin` may be here. In particular, the library `libm.so.*` may also be placed in `/usr/lib` if it is not required by anything in `/bin` or `/sbin`.
13. The usual placement of this binary is `/usr/bin/cpp`.
14. This is commonly used for 64-bit or 32-bit support on systems which support multiple binary formats, but require libraries of the same name. In this case, `/lib32` and `/lib64` might be the library directories, and `/lib` a symlink to one of them.
15. `/lib<qual>/cpp` is still permitted: this allows the case where `/lib` and `/lib<qual>` are the same (one is a symbolic link to the other).
16. A compliant implementation with two CDROM drives might have `/media/cdrom0` and `/media/cdrom1` with `/media/cdrom` a symlink to either of these.
17. If the home directory of the root account is not stored on the root partition it will be necessary to make certain it will default to `/` if it can not be located.

    We recommend against using the root account for tasks that can be performed as an unprivileged user, and that it be used solely for system administration. For this reason, we recommend that subdirectories for mail and other applications 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.
18. Originally, `/sbin` binaries were kept in `/etc`.
19. Deciding what things go into *"sbin"* directories is simple: if a normal (not a system administrator) user will ever run it directly, then it must be placed in one of the *"bin"* directories. Ordinary users should not have to place any of the `sbin` directories in their path.

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

    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.
20. This is particularly important as these areas will often contain both files initially installed by the distributor, and those added by the administrator.

Chapter 4. The /usr Hierarchy

4.1. Purpose

`/usr` is the second major section of the filesystem. `/usr` is shareable, read-only data. That means that `/usr` should be shareable between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.

Large software packages must not use a direct subdirectory under the `/usr` hierarchy.

4.2. Requirements

The following directories, or symbolic links to directories, are required in `/usr`.

Directory	Description
`bin`	Most user commands
`include`	Header files included by C programs
`lib`	Libraries
`local`	Local hierarchy (empty after main installation)
`sbin`	Non-vital system binaries
`share`	Architecture-independent data

4.3. Specific Options

Directory	Description
`X11R6`	XWindow System, version 11 release 6 (optional)
`games`	Games and educational binaries (optional)
`lib<qual>`	Alternate Format Libraries (optional)
`src`	Source code (optional)

An exception is made for the X Window System because of considerable precedent and widely-accepted practice.

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/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.

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

(optional)

4.4.1. Purpose

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 must be present if `/usr/X11R6` exists:

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

In general, software must 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.

4.4.2. Specific Options

Host-specific data in `/usr/X11R6/lib/X11` should be interpreted as a demonstration file. Applications requiring information about the current host must reference a configuration file in `/etc/X11`, which may be linked to a file in `/usr/X11R6/lib`. [1]

4.5. /usr/bin : Most user commands

4.5.1. Purpose

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

4.5.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/bin`, if the corresponding subsystem is installed:

Directory	Description
`mh`	Commands for the MH mail handling system (optional)

`/usr/bin/X11` must be a symlink to `/usr/X11R6/bin` if the latter exists.

The following files, or symbolic links to files, must be in `/usr/bin`, if the corresponding subsystem is installed:

Command	Description
perl	The Practical Extraction and Report Language (optional)
python	The Python interpreted language (optional)
tclsh	Simple shell containing Tcl interpreter (optional)
wish	Simple Tcl/Tk windowing shell (optional)

Command	Description
expect	Program for interactive dialog (optional)

Rationale

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. They may be symlinks to the physical location of the shell interpreters.

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

4.6.1. Purpose

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

4.6.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/include`, if the corresponding subsystem is installed:

Directory	Description
bsd	BSD compatibility include files (optional)

The symbolic link `/usr/include/X11` must link to `/usr/X11R6/include/X11` if the latter exists.

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

4.7.1. Purpose

`/usr/lib` includes object files, libraries, and internal binaries that are not intended to be executed directly by users or shell scripts. [2]

Applications may use a single subdirectory under `/usr/lib`. If an application uses a subdirectory, all architecture-dependent data exclusively used by the application must be placed within that subdirectory. [3]

4.7.2. Specific Options

For historical reasons, **/usr/lib/sendmail** must be a symbolic link to **/usr/sbin/sendmail** if the latter exists. [4]

If /lib/X11 exists, /usr/lib/X11 must be a symbolic link to /lib/X11, or to whatever /lib/X11 is a symbolic link to. [5]

4.8. /usr/lib<qual> : Alternate format libraries (optional)

4.8.1. Purpose

/usr/lib<qual> performs the same role as /usr/lib for an alternate binary format, except that the symbolic links /usr/lib<qual>/sendmail and /usr/lib<qual>/X11 are not required. [6]

4.8.2. /usr/local : Local hierarchy

4.8.2.1. Purpose

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.

Locally installed software must be placed within /usr/local rather than /usr unless it is being installed to replace or upgrade software in /usr. [7]

4.8.2.2. Requirements

The following directories, or symbolic links to directories, must be in /usr/local

Directory	Description
bin	Local binaries
etc	Host-specific system configuration for local binaries
games	Local game binaries
include	Local C header files
lib	Local libraries
man	Local online manuals
sbin	Local system binaries
share	Local architecture-independent hierarchy
src	Local source code

No other directories, except those listed below, may be in /usr/local after first installing a FHS-compliant system.

4.8.2.3. Specific Options

If directories /lib<qual> or /usr/lib<qual> exist, the equivalent directories must also exist in /usr/local.

/usr/local/etc may be a symbolic link to /etc/local.

Rationale

The consistency of `/usr/local/etc` is beneficial to installers, and is already used in other systems. As all of `/usr/local` needs to be backed up to reproduce a system, it introduces no additional maintenance overhead, but a symlink to `/etc/local` is suitable if systems want alltheir configuration under one hierarchy.

Note that `/usr/etc` is still not allowed: programs in `/usr` should place configuration files in `/etc`.

4.9. /usr/local/share

The requirements for the contents of this directory are the same as `/usr/share`. The only additional constraint is that `/usr/local/share/man` and `/usr/local/man` directories must be synonomous (usually this means that one of them must be a symbolic link). [8]

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

4.10.1. Purpose

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 must be placed in `/sbin` instead. [9]

4.11. /usr/share : Architecture-independent data

4.11.1. Purpose

The `/usr/share` hierarchy is for all read-only architecture independent data files. [10]

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.

Game data stored in `/usr/share/games` must be purely static data. Any modifiable files, such as score files, game play logs, and so forth, should be placed in `/var/games`.

4.11.2. Requirements

The following directories, or symbolic links to directories, must be in `/usr/share`

Directory	Description
`man`	Online manuals
`misc`	Miscellaneous architecture-independent data

4.11.3. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/share`, if the corresponding subsystem is installed:

Directory	Description
`dict`	Word lists (optional)
`doc`	Miscellaneous documentation (optional)
`games`	Static data files for /usr/games (optional)
`info`	GNU Info system s primary directory (optional)
`locale`	Locale information (optional)
`nls`	Message catalogs for Native language support (optional)
`sgml`	SGML data (optional)
`terminfo`	Directories for terminfo database (optional)
`tmac`	troff macros not distributed with groff (optional)
`xml`	XML data (optional)
`zoneinfo`	Timezone information and configuration (optional)

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.

4.11.4. /usr/share/dict : Word lists (optional)

4.11.4.1. Purpose

This directory is the home for word lists on the system; 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.

Rationale

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

4.11.4.2. Specific Options

The following files, or symbolic links to files, must be in `/usr/share/dict`, if the corresponding subsystem is installed:

File	**Description**
`words`	List of English words (optional)

Sites that require both American and British spelling 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 must be included here, if present.

4.11.5. /usr/share/man : Manual pages

4.11.5.1. Purpose

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

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. [11]

Manual pages are stored in `<mandir>/<locale>/man<section>/<arch>`. An explanation of `<mandir>`, `<locale>`, `<section>`, and `<arch>` is given below.

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 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 Programs used by system administrators for system operation and maintenance are documented here. Some of these programs are also occasionally useful for normal users.

4.11.5.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/share/<mandir>/<locale>`, unless they are empty: [12]

Directory	Description
man1	User programs (optional)
man2	System calls (optional)
man3	Library calls (optional)
man4	Special files (optional)
man5	File formats (optional)
man6	Games (optional)
man7	Miscellaneous (optional)
man8	System administration (optional)

The component `<section>` describes the manual section.

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 must be taken from ISO 639 (a code for the representation of names of languages). It must be two characters wide and specified with lowercase letters only.

The `<territory>` field must 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.) It must be two characters wide and specified with uppercase letters only. [13]

The `<character-set>` field must 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<section>` 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

Language	Territory	Character Set	Directory
English	United Kingdom	ISO 8859-15	/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 documentation on device-drivers or low-level system administration commands. These must be placed under an `<arch>` directory in the appropriate `man<section>` directory; for example, a man page for the i386 ctrlaltdel(8) command might be placed in `/usr/share/man/<locale>/man8/i386/ctrlaltdel.8`.

Manual pages for commands and data under `/usr/local` are stored in `/usr/local/man`. Manual pages for X11R6 are stored in `/usr/X11R6/man`. It follows that all manual page hierarchies in the system must have the same structure as `/usr/share/man`.

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

The numbered sections "1" through "8" are traditionally defined. In general, the file name for manual pages located within a particular section end with `.<section>`.

In addition, some large sets of application-specific manual pages have an additional suffix appended to the manual page filename. For example, the MH mail handling system manual pages must have `mh` appended to all MH manuals. All X Window System manual pages must 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.)

4.11.6. /usr/share/misc : Miscellaneous architecture-independent data

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

4.11.6.1. Specific Options

The following files, or symbolic links to files, must be in `/usr/share/misc`, if the corresponding subsystem is installed:

File	Description
`ascii`	ASCII character set table (optional)

File	Description
`magic`	Default list of magic numbers for the file command (optional)
`termcap`	Terminal capability database (optional)
`termcap.db`	Terminal capability database (optional)

Other (application-specific) files may appear here, but a distributor may place them in `/usr/lib` at their discretion. [14]

4.11.7. /usr/share/sgml : SGML data (optional)

4.11.7.1. Purpose

`/usr/share/sgml` contains architecture-independent files used by SGML applications, such as ordinary catalogs (not the centralized ones, see `/etc/sgml`), DTDs, entities, or style sheets.

4.11.7.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/share/sgml`, if the corresponding subsystem is installed:

Directory	Description
`docbook`	docbook DTD (optional)
`tei`	tei DTD (optional)
`html`	html DTD (optional)
`mathml`	mathml DTD (optional)

Other files that are not specific to a given DTD may reside in their own subdirectory.

4.11.8. /usr/share/xml : XML data (optional)

4.11.8.1. Purpose

`/usr/share/xml` contains architecture-independent files used by XML applications, such as ordinary catalogs (not the centralized ones, see `/etc/sgml`), DTDs, entities, or style sheets.

4.11.8.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/share/xml`, if the corresponding subsystem is installed:

Directory	Description
`docbook`	docbook XML DTD (optional)
`xhtml`	XHTML DTD (optional)

Directory	Description
`mathml`	MathML DTD (optional)

4.12. /usr/src : Source code (optional)

4.12.1. Purpose

Source code may be place placed in this subdirectory, only for reference purposes. [15]

Notes

1. Examples of such configuration files include `Xconfig`, `XF86Config`, or `system.twmrc`)
2. Miscellaneous architecture-independent application-specific static files and subdirectories must be placed in `/usr/share`.
3. For example, the `perl5` subdirectory for Perl 5 modules and libraries.
4. Some executable commands such as **makewhatis** and **sendmail** have also been traditionally placed in `/usr/lib`. **makewhatis** is an internal binary and must be placed in a binary directory; users access only **catman**. Newer **sendmail** binaries are now placed by default in `/usr/sbin`. Additionally, systems using a *sendmail*-compatible mail transfer agent must provide **/usr/sbin/sendmail** as a symbolic link to the appropriate executable.
5. Host-specific data for the X Window System must not be stored in `/usr/lib/X11`. Host-specific configuration files such as `Xconfig` or `XF86Config` must be stored in `/etc/X11`. This includes 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`).
6. The case where `/usr/lib` and `/usr/lib<qual>` are the same (one is a symbolic link to the other) these files and the per-application subdirectories will exist.
7. 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 must not be placed outside of `/usr/local` without good reason.
8. `/usr/local/man` may be deprecated in future FHS releases, so if all else is equal, making that one a symlink seems sensible.
9. Locally installed system administration programs should be placed in `/usr/local/sbin`.
10. Much of this data originally lived in `/usr` (`man`, `doc`) or `/usr/lib` (`dict`, `terminfo`, `zoneinfo`).
11. Obviously, there are no manual pages in `/` because they are not required at boot time nor are they required in emergencies. Really.
12. For example, if `/usr/local/man` has no manual pages in section 4 (Devices), then `/usr/local/man/man4` may be omitted.
13. A major exception to this rule is the United Kingdom, which is ‘GB’ in the ISO 3166, but ‘UK’ for most email addresses.

14. 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`
15. Generally, source should not be built within this hierarchy.

Chapter 5. The /var Hierarchy

5.1. Purpose

/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 may be shared, 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 must 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 must 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.

5.2. Requirements

The following directories, or symbolic links to directories, are required in /var.

Directory	Description
cache	Application cache data
lib	Variable state information
local	Variable data for /usr/local
lock	Lock files
log	Log files and directories
opt	Variable data for /opt
run	Data relevant to running processes
spool	Application spool data
tmp	Temporary files preserved between system reboots

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

```
/var/backups
/var/cron
/var/msgs
/var/preserve
```

5.3. Specific Options

The following directories, or symbolic links to directories, must be in /var, if the corresponding subsystem is installed:

Directory	Description
`account`	Process accounting logs (optional)
`crash`	System crash dumps (optional)
`games`	Variable game data (optional)
`mail`	User mailbox files (optional)
`yp`	Network Information Service (NIS) database files (optional)

5.4. /var/account : Process accounting logs (optional)

5.4.1. Purpose

This directory holds the current active process accounting log and the composite process usage data (as used in some UNIX-like systems by **lastcomm** and **sa**).

5.5. /var/cache : Application cache data

5.5.1. Purpose

`/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 must 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 must always be able to recover from manual deletion of these files (generally because of a disk space shortage). No other requirements are made on the data format of the cache directories.

Rationale

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

5.5.2. Specific Options

Directory	Description
`fonts`	Locally-generated fonts (optional)
`man`	Locally-formatted manual pages (optional)

Directory	Description
www	WWW proxy or cache data (optional)
<package>	Package specific cache data (optional)

5.5.3. /var/cache/fonts : Locally-generated fonts (optional)

5.5.3.1. Purpose

The directory /var/cache/fonts should be used to store any dynamically-created fonts. In particular, all of the fonts which are automatically generated by **mktexpk** must be located in appropriately-named subdirectories of /var/cache/fonts. [1]

5.5.3.2. Specific Options

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

5.5.4. /var/cache/man : Locally-formatted manual pages (optional)

5.5.4.1. Purpose

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<section> 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 <path>/man/<locale>/man<section>, the directory to place formatted man pages in is /var/cache/man/<catpath>/<locale>/cat<section>, where <catpath> is derived from <path> by removing any leading usr and/or trailing share pathname components. (Note that the <locale> component may be missing.) [2]

Man pages written to /var/cache/man may eventually be transferred to the appropriate preformatted directories in the source man hierarchy or expired; likewise formatted man pages in the source man hierarchy 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 must be installed in the source man hierarchy (e.g. /usr/share/man/cat<section>). /var/cache/man is reserved as a writable cache for formatted manual pages.

Rationale

Release 1.2 of the standard specified `/var/catman` for this hierarchy. The path has been moved under `/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.

5.6. /var/crash : System crash dumps (optional)

5.6.1. Purpose

This directory holds system crash dumps. As of the date of this release of the standard, system crash dumps were not supported under Linux but may be supported by other systems which may comply with the FHS.

5.7. /var/games : Variable game data (optional)

5.7.1. Purpose

Any variable data relating to games in `/usr` 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, must remain elsewhere, such as `/usr/share/games`.

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/lib`. Additionally, `/var/games` is the path traditionally used by BSD.

5.8. /var/lib : Variable state information

5.8.1. Purpose

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 must never need to modify files in `/var/lib` 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) must use a subdirectory of /var/lib for its data. There is one required subdirectory, /var/lib/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. [3]

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

5.8.2. Requirements

The following directories, or symbolic links to directories, are required in /var/lib:

Directory	Description
misc	Miscellaneous state data

5.8.3. Specific Options

The following directories, or symbolic links to directories, must be in /var/lib, if the corresponding subsystem is installed:

Directory	Description
<editor>	Editor backup files and state (optional)
<pkgtool>	Packaging support files (optional)
<package>	State data for packages and subsystems (optional)
hwclock	State directory for hwclock (optional)
xdm	X display manager variable data (optional)

5.8.4. /var/lib/<editor> : Editor backup files and state (optional)

5.8.4.1. Purpose

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/lib (for example, GNU Emacs would place lock files in /var/lib/emacs/lock).

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

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/lib`.

5.8.5. /var/lib/hwclock : State directory for hwclock (optional)

5.8.5.1. Purpose

This directory contains the file `/var/lib/hwclock/adjtime`.

Rationale

In FHS 2.1, this file was `/etc/adjtime`, but as **hwclock** updates it, that was obviously incorrect.

5.8.6. /var/lib/misc : Miscellaneous variable data

5.8.6.1. Purpose

This directory contains variable data not placed in a subdirectory in `/var/lib`. An attempt should be made to use relatively unique names in this directory to avoid namespace conflicts. [4]

5.9. /var/lock : Lock files

5.9.1. Purpose

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

Lock files for devices and other resources shared by multiple applications, 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/ttyS0 the file "LCK..ttyS0" would be created. [5]

The format used for the contents of such 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.

5.10. /var/log : Log files and directories

5.10.1. Purpose

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

5.10.2. Specific Options

The following files, or symbolic links to files, must be in `/var/log`, if the corresponding subsystem is installed:

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

5.11. /var/mail : User mailbox files (optional)

5.11.1. Purpose

The mail spool must be accessible through `/var/mail` and the mail spool files must take the form `<username>`.[6]

User mailbox files in this location must be stored in the standard UNIX mailbox format.

Rationale

The logical location for this directory was changed from `/var/spool/mail` in order to bring FHS in-line with nearly every UNIX implementation. This change is important for inter-operability since a single `/var/mail` is often shared between multiple hosts and multiple UNIX implementations (despite NFS locking issues).

It is important to note that there is no requirement to physically move the mail spool to this location. However, programs and header files must be changed to use `/var/mail`.

5.12. /var/opt : Variable data for /opt

5.12.1. Purpose

Variable data of the packages in `/opt` must be installed in `/var/opt/<subdir>`, where `<subdir>` 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/<subdir>`.

Rationale

Refer to the rationale for `/opt`.

5.13. /var/run : Run-time variable data

5.13.1. Purpose

This directory contains system information data describing the system since it was booted. Files under this directory must be cleared (removed or truncated as appropriate) at the beginning of the boot process. Programs may have a subdirectory of `/var/run`; this is encouraged for programs that use more than one run-time file. [7] Process identifier (PID) files, which were originally placed in `/etc`, must 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`.

5.13.2. Requirements

The internal format of PID files remains unchanged. The file must 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.

System programs that maintain transient UNIX-domain sockets must place them in this directory.

5.14. /var/spool : Application spool data

5.14.1. Purpose

`/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. [8]

5.14.2. Specific Options

The following directories, or symbolic links to directories, must be in `/var/spool`, if the corresponding subsystem is installed:

Directory	Description
`lpd`	Printer spool directory (optional)
`mqueue`	Outgoing mail queue (optional)
`news`	News spool directory (optional)
`rwho`	Rwhod files (optional)
`uucp`	Spool directory for UUCP (optional)

5.14.3. /var/spool/lpd : Line-printer daemon print queues (optional)

5.14.3.1. Purpose

The lock file for **lpd**, `lpd.lock`, must 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`.

5.14.3.2. Specific Options

Directory	Description
`printer`	Spools for a specific printer (optional)

5.14.4. /var/spool/rwho : Rwhod files (optional)

5.14.4.1. Purpose

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

Rationale

Some BSD releases use `/var/rwho` for this data; given its historical location in `/var/spool` on other systems and its approximate fit to the definition of 'spooled' data, this location was deemed more appropriate.

5.15. /var/tmp : Temporary files preserved between system reboots

5.15.1. Purpose

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`.

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

5.16.1. Purpose

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

Rationale

`/var/yp` is the standard directory for NIS (YP) data and is almost exclusively used in NIS documentation and systems. [9]

Notes

1. This standard does not currently incorporate the TeX Directory Structure (a document that describes the layout TeX files and directories), but it may be useful reading. It is located at ftp://ctan.tug.org/tex/
2. 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`.
3. An important difference between this version of this standard and previous ones is that applications are now required to use a subdirectory of `/var/lib`.
4. This hierarchy should contain files stored in `/var/db` in current BSD releases. These include `locate.database` and `mountdtab`, and the kernel symbol database(s).
5. Then, anything wishing to use `/dev/ttyS0` can read the lock file and act accordingly (all locks in `/var/lock` should be world-readable).
6. Note that `/var/mail` may be a symbolic link to another directory.
7. `/var/run` should be unwritable for unprivileged users (root or users running daemons); it is a major security problem if any user can write in this directory.
8. UUCP lock files must be placed in `/var/lock`. See the above section on `/var/lock`.
9. NIS should not be confused with Sun NIS+, which uses a different directory, `/var/nis`.

Chapter 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. / : Root directory

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

Linux systems which require them place these additional files into `/bin`:

- **setserial**

6.1.3. /dev : Devices and special files

The following devices must exist under /dev.

`/dev/null`

All data written to this device is discarded. A read from this device will return an EOF condition.

`/dev/zero`

This device is a source of zeroed out data. All data written to this device is discarded. A read from this device will return as many bytes containing the value zero as was requested.

`/dev/tty`

This device is a synonym for the controlling terminal of a process. Once this device is opened, all reads and writes will behave as if the actual controlling terminal device had been opened.

Rationale

Previous versions of the FHS had stricter requirements for `/dev`. Other devices may also exist in /dev. Device names may exist as symbolic links to other device nodes located in /dev or subdirectories of /dev. There is no requirement concerning major/minor number values.

6.1.4. /etc : Host-specific system configuration

Linux systems which require them place these additional files into /etc.

- lilo.conf

6.1.5. /lib64 and /lib32 : 64/32-bit libraries (architecture dependent)

The 64-bit architectures PPC64, s390x, sparc64 and AMD64 must place 64-bit libraries in /lib64, and 32-bit (or 31-bit on s390) libraries in /lib.

The 64-bit architecture IA64 must place 64-bit libraries in /lib.

Rationale

This is a refinement of the general rules for /lib<qual> and /usr/lib<qual>. The architectures PPC64, s390x, sparc64 and AMD64 support support both 32-bit (for s390 more precise 31-bit) and 64-bit programs. Using lib for 32-bit binaries allows existing binaries from the 32-bit systems to work without any changes: such binaries are expected to be numerous. IA-64 uses a different scheme, reflecting the deprecation of 32-bit binaries (and hence libraries) on that architecture.

6.1.6. /proc : Kernel and process information virtual filesystem

The proc filesystem is 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.

6.1.7. /sbin : Essential system binaries

Linux systems place these additional files into /sbin.

- Second extended filesystem commands (optional):
  - **badblocks**
  - **dumpe2fs**
  - **e2fsck**
  - **mke2fs**
  - **mklost+found**
  - **tune2fs**

- Boot-loader map installer (optional):
  - **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.

6.1.8. /usr/include : Header files included by C programs

These symbolic links are required if a C or C++ compiler is installed and only for systems not based on glibc.

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

6.1.9. /usr/src : Source code

For systems based on glibc, there are no specific guidelines for this directory. For systems based on Linux libc revisions prior to glibc, the following guidelines and rationale apply:

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 must be located in these directories:

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

`<arch>` is the name of the system architecture.

Note

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

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.

6.1.10. /var/spool/cron : cron and at jobs

This directory contains the variable data for the **cron** and **at** programs.

Chapter 7. Appendix

7.1. The FHS mailing list

The FHS mailing list is located at <freestandards-fhs-discuss@lists.sourceforge.net>. You can subscribe to the mailing list at this page http://sourceforge.net/projects/freestandards/.

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.

7.2. 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 with the help of 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 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.

7.3. General Guidelines

Here are some of the guidelines that have been used in the development of this standard:

- Solve technical problems while limiting transitional difficulties.
- Make the specification reasonably stable.
- Gain the approval of distributors, developers, and other decision-makers in relevant development groups and encourage their participation.
- Provide a standard that is attractive to the implementors of different UNIX-like systems.

7.4. 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 than previous attempts at filesystem hierarchy standardization, periodic updates may become necessary as requirements change in relation to emerging technology. It is also possible that better solutions to the problems addressed here will be discovered so that our solutions will no longer be the best possible solutions. Supplementary drafts may be released 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 welcome. Any comments or suggestions for changes may be directed to the FHS editor (Daniel Quinlan <quinlan@pathname.com>) or the FHS mailing list. Typographical or grammatical comments should be directed to the FHS editor.

Before sending mail to the mailing list it is requested that you first contact the FHS editor in order to avoid excessive re-discussion of old topics.

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.

7.5. Acknowledgments

The developers of the FHS wish to thank the developers, system administrators, and users whose input was essential to this standard. We wish to thank each of the contributors who helped to write, compile, and compose this standard.

The FHS Group also wishes to thank those Linux developers who supported the FSSTND, the predecessor to this standard. If they hadn't demonstrated that the FSSTND was beneficial, the FHS could never have evolved.

7.6. Contributors

Brandon S. Allbery	<bsa@kf8nh.wariat.org>
Keith Bostic	<bostic@cs.berkeley.edu>
Drew Eckhardt	<drew@colorado.edu>
Rik Faith	<faith@cs.unc.edu>
Stephen Harris	<sweh@spuddy.mew.co.uk>
Ian Jackson	<ijackson@cus.cam.ac.uk>
Andreas Jaeger	<aj@suse.de>
John A. Martin	<jmartin@acm.org>
Ian McCloghrie	<ian@ucsd.edu>
Chris Metcalf	<metcalf@lcs.mit.edu>
Ian Murdock	<imurdock@debian.org>
David C. Niemi	<niemidc@clark.net>
Daniel Quinlan	<quinlan@pathname.com>
Eric S. Raymond	<esr@thyrsus.com>
Rusty Russell	<rusty@rustcorp.com.au>

Mike Sangrey	<mike@sojurn.lns.pa.us>
David H. Silber	<dhs@glowworm.firefly.com>
Thomas Sippel-Dau	<t.sippel-dau@ic.ac.uk>
Theodore Ts'o	<tytso@athena.mit.edu>
Stephen Tweedie	<sct@dcs.ed.ac.uk>
Fred N. van Kempen	<waltje@infomagic.com>
Bernd Warken	<bwarken@mayn.de>
Christopher Yeoh	<cyeoh@samba.org>