Learn Linux, 101: Find and place system files

Where things go in the Filesystem Hierarchy Standard

Ian Shields
Senior Programmer
IBM

09 June 2010

Learn the correct location for files under the Filesystem Hierarchy Standard (FHS) on a Linux® system and learn how to find important files and commands. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to explore file organization and management.

View more content in this series

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our developerWorks roadmap for LPIC-1 for a description of and link to each article in this series. The roadmap is in progress and reflects the latest (April 2009) objectives for the LPIC-1 exams: as we complete articles, we add them to the roadmap. In the meantime, though, you can find earlier versions of similar material, supporting previous LPIC-1 objectives prior to April 2009, in our LPI certification exam prep tutorials.

Overview

In this article, learn about the Filesystem Hierarchy Standard (FHS). Learn to:

- Recognize where to place files under the FHS
- Find files and commands on your Linux system
- Find other important files and directories defined in the FHS and understand their purposes

This article helps you prepare for Objective 104.7 in Topic 104 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 2.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here. In particular, much of the output we show is highly dependent on the packages that are already installed on our systems. Your own output may be quite different, although you should be able to recognize the important commonalities.

Filesystem Hierarchy Standard

Connect with Ian

Ian is one of our most popular and prolific authors. Browse all of Ian's articles on developerWorks. Check out Ian's profile and connect with him, other authors, and fellow readers in My developerWorks.

The Filesystem Hierarchy Standard is a document that specifies a common layout of directories on a Linux or other UNIX®-like system. By placing files in the same general place across Linux distributions, the FHS simplifies distribution-independent software development. The FHS is also used in the Linux Standard Base (see Resources). The FHS allows both users and software to predict the location of installed files and directories. An FHS-compliant filesystem assumes that the operating system supports the basic security features found in most UNIX filesystems.

The two independent FHS categories

At the core of the FHS are two independent characteristics of files:

Shareable vs. unshareable
Shareable files can be located on one system and used on another, while unshareable files must reside on the system on which they are used.

Static vs. variable
Static files change only through system administrator intervention, such as installing or upgrading a package, and include documentation, libraries, and binaries. Variable files are all other files, such as logs, spool files, databases, and user data, which are subject to change by users and by system processes.

These distinctions allow files with different sets of characteristics to be stored on different filesystems. Table 1 is an example from the FHS document showing a layout that would be FHS compliant.

Table 1. FHS example

	Shareable	Unshareable
Static	/usr /opt	/etc /boot
Variable	/var/mail /var/spool/news	/var/run /var/lock

Where's that file?

Linux systems often contain hundreds of thousands of files. A 64-bit Fedora 13 system that I recently installed has over 75,000 files in the /usr hierarchy alone. Most of my other installations have over 100,000 files and often 200,000 files or more. The next four sections look at tools to help you find files, particularly programs, in this vast sea of data.

What's on your PATH

If you have used several Linux systems, you may have noticed that if you log in as root, you are able to execute commands such as `fdisk`, which you cannot execute if you are a user. If you run a program at the command line, the bash (or other) shell searches through a list of directories to find the program you requested. The list of directories is specified in your PATH environment variable, and root's path may include /sbin, while non-root user paths do not. Listing 1 shows user path examples from two different distributions, as well as a root path example.

Listing 1. Some PATH examples

```
ian@pinguino:~$ # An Ubuntu 9.10 system
ian@pinguino:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

[ian@echidna ~]$ # An openSUSE 11.2 system
ian@attic4:~> echo $PATH
/usr/lib64/mpi/gcc/openmpi/bin:/home/ian/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/u
sr/X11R6/bin:/usr/games:/usr/lib64/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin

[root@echidna ~]# # And as root
attic4:~ # echo $PATH
/usr/lib64/mpi/gcc/openmpi/bin:/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/local/bin:/
usr/bin:/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/games:/usr/lib64/jvm/jre/bin:/usr/lib/mit/b
in:/usr/lib/mit/sbin
```

As you can see, the PATH variable is just a list of directory names, separated by colons. Since the `fdisk` command is actually located in /sbin/fdisk, only the first and last of these paths would allow the user to run it by typing `fdisk` without providing a fully qualified name (/sbin/fdisk).

Usually, your path is set in an initialization file such as .bash_profile or .bashrc. You can change it for the current bash process by specifying a new path. Remember to export the PATH variable if you want the new value to be available to other processes that you start from this one. An example is shown in Listing 2.

Listing 2. Changing your PATH

```
ian@attic4:~> fdisk
Absolute path to 'fdisk' is '/sbin/fdisk', so running it may require superuser privileges
 (e.g. root).
ian@attic4:~> export PATH=/sbin:$PATH
ian@attic4:~> fdisk

Usage: fdisk [-l] [-b SSZ] [-u] device
E.g.: fdisk /dev/hda  (for the first IDE disk)
  or: fdisk /dev/sdc  (for the third SCSI disk)
  or: fdisk /dev/eda  (for the first PS/2 ESDI drive)
  or: fdisk /dev/rd/c0d0  or: fdisk /dev/ida/c0d0  (for RAID devices)
```

The which, type, and whereis commands

In the previous section, you saw why the `fdisk` command might not be available if you attempted to run it. However, there are several other useful commands that can help you find which command would in fact run if you typed a command name.

The which command

You can use the `which` command to search your path and find out which command will be executed (if any) when you type a command. Listing 3 shows an example of finding the `fdisk` command.

Listing 3. Using which

```
ian@attic4:~> which fdisk
which: no fdisk in (/usr/lib64/mpi/gcc/openmpi/bin:/home/ian/bin:/usr/local/bin:/usr/bin:
/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/games:/usr/lib64/jvm/jre/bin:/usr/lib/mit/bin:/usr/
lib/mit/sbin)
ian@attic4:~> export PATH=/sbin:$PATH
ian@attic4:~> which fdisk
/sbin/fdisk
```

The `which` command shows you the first occurrence of a command in your path. If you want to know if there are multiple occurrences, then add the `-a` option as shown in Listing 4.

Listing 4. Using which to find multiple occurrences

```
ian@attic4:~> which awk
/usr/bin/awk
ian@attic4:~> which -a awk
/usr/bin/awk
/bin/awk
/usr/bin/X11/awk
```

Here we find the `awk` command in three places: in /usr/bin (which is the main directory for commands on the system), in /bin (which contains commands that may be used both by the system administrator and by users, but which are required when no other filesystems are mounted), and also in /usr/bin/X11 (which contains the binaries for the X window system).

Another article in this series, "Learn Linux 101: Create and change hard and symbolic links," shows you how to check that these three different files all eventually represent the same underlying `gawk` command as shown in Listing 5.

Listing 5. Awk commands lead to gawk

```
ian@attic4:~> ls -l $(which -a awk)
lrwxrwxrwx 1 root root 4 2010-02-09 00:46 /bin/awk -> gawk
lrwxrwxrwx 1 root root 8 2010-02-09 00:46 /usr/bin/awk -> /bin/awk
lrwxrwxrwx 1 root root 8 2010-02-09 00:46 /usr/bin/X11/awk -> /bin/awk
```

The type command

There are some commands that the `which` command will not find, such as shell builtins. The `type` command is a builtin that will tell you how a given command string will be evaluated for execution. Listing 6 uses `which` and `type` to show that the `type` command is not an executable found on your path, but is a shell builtin.

Listing 6. Using type

```
ian@attic4:~> which type
which: no type in (/usr/lib64/mpi/gcc/openmpi/bin:/home/ian/bin:/usr/local/bin:/usr/bin:/
bin:/usr/bin/X11:/usr/X11R6/bin:/usr/games:/usr/lib64/jvm/jre/bin:/usr/lib/mit/bin:/usr/l
ib/mit/sbin)
ian@attic4:~> type type
type is a shell builtin
```

The whereis command

If you want more information than just the location of a program, then you can use the `whereis` command. For example, you can find the man pages or other information, as shown in Listing 7.

Listing 7. Using whereis to find man pages

```
ian@attic4:~> whereis awk
awk: /bin/awk /usr/bin/awk /usr/lib64/awk /usr/bin/X11/awk /usr/share/awk
/usr/share/man/man1/awk.1.gz /usr/share/man/man1p/awk.1p.gz
```

Note that the copy of `awk` in /sbin was not found by `whereis`. The directories used by `whereis` are fixed, so the command may not always find what you are looking for. The `whereis` command can also search for source files, specify alternate search paths, and search for unusual entries. Consult the man pages to see how to override this behavior or change the fixed paths used by `whereis`.

The find command

In an earlier article in this series, "Learn Linux 101: File and directory management," you learned how to find files based on name (including wildcards), path, size, or timestamp. In another earlier article in this series, "Learn Linux 101: Create and change hard and symbolic links," you learned how to find the links to a particular file or inode.

The `find` command is the Swiss Army knife of file searching tools on Linux systems. Two other capabilities that you may find useful are its ability to find files based on user or group name and its ability to find files based on permissions.

Suppose you want to see what files a user has in the /tmp hierarchy. Listing 8 shows how the root user could find all the files for user ian in in /tmp.

Listing 8. Finding files by user and group

```
attic4:~ # find /tmp -user ian
/tmp/kde-ian
/tmp/kde-ian/closeditems
/tmp/kde-ian/closeditems/_1.66
/tmp/kde-ian/systemsettingsR27913.tmp
/tmp/.ICE-unix/2288
/tmp/orbit-ian
/tmp/orbit-ian/linc-12f7-0-33cb4ce9b1fbf
/tmp/orbit-ian/linc-7d00-0-70e5ebaa4ddac
/tmp/orbit-ian/linc-12ea-0-68260abbd2051
/tmp/orbit-ian/linc-12ea-0-3377ca55c0bd2
/tmp/ksocket-ian
/tmp/ksocket-ian/klauncherMT2183.slave-socket
...
```

You can also find files by group using the `-group` test. And you can find files that do not belong to any user or group on the system using the `-nouser` and `-nogroup` options. As with other tests, you can negate the test using `!`. I usually set my user number to 1000, as that is the default on some systems. I also create a group called ian with 1000 as the group number. Other systems still start at 500, or put new users in the group 'users' by default. Some of my older research material that was archived from a Red Hat 6.2 system still has user 500. Listing 9 shows how to find some directories that are not owned by my current user group. The research/rh62/involution is owned by user 500 and group 4, neither of which exist on my current system. To find files or directories by numeric user id or group id, use the `-uid` or `-gid` tests.

Listing 9. Finding directories not owned by ian

```
ian@attic4:~> find -L research -maxdepth 2 -type d ! -group ian
research/rh62/involution
research/rh62/programs
research/lost+found
find: `research/lost+found': Permission denied
ian@attic4:~> ls -ld research/rh62/involution
drwxr-xr-x. 2 500 4 4096 1999-11-10 08:09 research/rh62/involution
```

To find files by permission, you can use the `-perm` test along with symbolic expressions similar to those used with the `chmod` or `umask` commands. You can search for exact permissions, but it is often more useful to prefix the permission expression with a hyphen to indicate that you want files with those permissions set, but that you don't care about other permissions. Listing 10 illustrates how to find files that are executable by user, group, and everyone, and two different ways of finding files that are not readable by others.

Listing 10. Finding files by permission

```
ian@attic4:~> find . -maxdepth 1 -type f -perm -uga=x
./.xinitrc.template
ian@attic4:~> ls -l ./.xinitrc.template
-rwxr-xr-x 1 ian users 1446 2010-02-09 08:55 ./.xinitrc.template
ian@attic4:~> find . -maxdepth 1 ! -perm  -o=r
./.Xauthority
./.pulse
...
ian@attic4:~> find . -maxdepth 1 ! -perm  -0004
./.Xauthority
./.pulse
...
```

We have covered several major types of search that you can do with the `find` command. To further narrow your output, you can combine multiple expressions, and you can add regular expressions to the mix. To learn more about this versatile command, use the man page, or better, use `info find` if you have the info system installed.

Listing 11 shows a final example of searching with `find`. This example does a `cd` to /usr/include to keep the listing length manageable, then finds all files containing `packet` in their path name without regard to case. The second example further restricts this output to files that are not directories and that are at least 1500 bytes in size. Actual output on your system may differ depending on which packages you have installed.

Listing 11. A final example of find

```
ian@attic4:/usr/include> find . -iregex ".*packet.*"
./c++/4.4/java/net/DatagramPacket.h
./c++/4.4/gnu/classpath/jdwp/processor/PacketProcessor.h
./c++/4.4/gnu/classpath/jdwp/transport/JdwpPacket.h
./c++/4.4/gnu/classpath/jdwp/transport/JdwpReplyPacket.h
./c++/4.4/gnu/classpath/jdwp/transport/JdwpCommandPacket.h
./netpacket
./netpacket/packet.h
./net/if_packet.h
./linux/if_packet.h
ian@attic4:/usr/include> find . -iregex ".*packet.*" ! -type d -size +1500c
./c++/4.4/java/net/DatagramPacket.h
./c++/4.4/gnu/classpath/jdwp/transport/JdwpPacket.h
./netpacket/packet.h
./linux/if_packet.h
```

Note that the regular expression must match the full path returned by `find`, and remember the difference between regular expressions and wildcards.

The locate and updatedb commands

The `find` command searches all the directories you specify, every time you run it. To speed things up, you can use another command, `locate`, which uses a database of stored path information rather than searching the filesystem every time.

The locate command

The `locate` command searches for matching files in a database that is usually updated daily by a cron job.

The `locate` command matches against any part of a path name, not just the file name. Put the file name in single quotes *and* include at least one globbing character to match more precisely. Listing 12 shows how to find paths containing the string `bin/ls`, and shows two examples of using globbing characters to restrict the output.

Listing 12. Using locate to find paths and restrict output

```
ian@attic4:~> locate /bin/ls
/bin/ls
/bin/lsmod
/usr/bin/lsattr
/usr/bin/lsb_release
/usr/bin/lscpu
/usr/bin/lsdev
/usr/bin/lshal
/usr/bin/lsof
/usr/bin/lsscsi
/usr/bin/lsusb
ian@attic4:~> locate '\/bin/ls'
/bin/ls
ian@attic4:~> locate '/bin/ls*'
/bin/ls
/bin/lsmod
```

The updatedb command

The default database used by `locate` is stored in the /var filesystem, in a location such as /var/lib/locatedb. This may be different on systems that use slocate or mlocate packages to provide additional security or speed. You can find statistics on your locate database using `locate -S` as shown in Listing 13.

Listing 13. Locatedb statistics

```
ian@attic4:~> locate -S
Database /var/lib/locatedb is in the GNU LOCATE02 format.
Locate database size: 3011297 bytes
All Filenames: 259149
File names have a cumulative length of 15751703 bytes.
Of those file names,

        11421 contain whitespace,
        0 contain newline characters,
        and 0 contain characters with the high bit set.
        Compression ratio 80.88% (higher is better)
```

The database is created or updated using the `updatedb` command. This is usually run daily as a cron job. The file /etc/updatedb.conf, or sometimes /etc/sysconfig/locate, is the configuration file for `updatedb`. To enable daily updates, the root user needs to edit /etc/updatedb.conf and set DAILY_UPDATE=yes. To create the database immediately, run the `updatedb` command as root.

Other considerations for using `locate` include security considerations and network file I/O considerations for daily builds of the `updatedb` database. Check the man pages and `updatedb` configuration files for more details.

FHS directories in the root filesystem

The FHS goal is to keep the root filesystem as small as possible. However, it must contain all the files necessary to boot, restore, recover, or repair the system, including the utilities that an experienced administrator would need for these tasks. Note that booting a system requires that enough files be on the root filesystem to permit mounting of other filesystems.

Table 2 shows the purpose of the directories that the FHS requires in the root (or /) filesystem. Either the directory or a symbolic link to it must be present, except for those marked as optional, which are required only if the corresponding subsystem is present.

Table 2. FHS root filesystem

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

/usr and /var hierarchies

The /usr and /var hierarchies are complex enough to have complete sections of the FHS devoted to them. The /usr filesystem is the second major section of the filesystem, containing shareable, read-only data. It can be shared between systems, although present practice does not often do this.

The /var filesystem contains variable data files, including spool directories and files, administrative and logging data, and transient and temporary files. Some portions of /var are not shareable between different systems, but others, such as /var/mail, /var/cache/man, /var/cache/fonts, and /var/spool/news, may be shared.

To fully understand the standard, read the FHS document (see Resources).

Resources

Learn

- Use the developerWorks roadmap for LPIC-1 to find the developerWorks articles to help you study for LPIC-1 certification based on the April 2009 objectives.
- At the LPIC Program site, find detailed objectives, task lists, and sample questions for the three levels of the Linux Professional Institute's Linux system administration certification. In particular, see their April 2009 objectives for LPI exam 101 and LPI exam 102. Always refer to the LPIC Program site for the latest objectives.
- Review the entire LPI exam prep series on developerWorks to learn Linux fundamentals and prepare for system administrator certification based on earlier LPI exam objectives prior to April 2009.
- Visit the home of the Filesystem Hierarchy Standard (FHS).
- Learn about the Linux Standard Base (LSB), a Free Standards Group (FSG) project to develop a standard binary operating environment.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

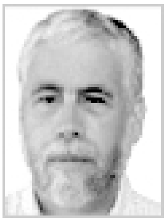
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Ian Shields



Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. Learn more about Ian in Ian's profile on developerWorks Community.