Filesystem Hierarchy Standard version 2.0

Daniel Quinlan
quinlan@transmeta.com

Transmeta Corporation

This talk will cover

- Introduction
- Basic theory and philosophy of Unix filesystem
- Changes in the new standard
- The Future
- Follow-up questions

What is the FHS?

- Requirements and guidelines for file and directory placement under Linux and Linux-like systems (such as Unix).
- Also describes the contents of some system files.
- Builds on the Linux Filesystem Standard (FSSTND). Last version was 1.2.

Why was a standard needed?

- There is no central authority for Linux
- People had different ideas about where to put files, directories, and the contents of some files that must be shared between applications.
- No applicable standards such as POSIX
- Unix design: programs don't know from where they are executed, pathnames are relatively fixed.
- Not a real problem today because of FSSTND and GNU coding standards.

So, what does FHS do for us?

- Gives us more source, binary, and documentation compatibility between Linux distributions.
- Can write scripts and administration tools for everyone.
- Limit number of places that a particular type of data can be located.

Background & History

- Early attempt at standardization
- State in 1993
  - binaries in /etc, no /sbin
  - set of binaries in /bin is large and haphazard
  - no /var
  - /usr/spool
  - /etc/lilo (binaries, configuration, & boot data)
  - /usr/spool/locks, etc.
- After release, followed by Debian, RedHat, and others

Why the new name?

- I didn't like the old name.
- The new standard can be used by Linux-like operating systems now.
- Major new features.

Sources for FHS 2.0

- FSSTND 1.2
- 4.4 BSD
- GNU applications and GNU coding standards
- Solaris 2, IRIX, HP-UX, etc.

Changing Standards

- Try to limit transitional difficulties (don't muck with it too much!)
- Open-standards
- Try to stick to what works, not what you think should work.

Who should care?

- Anyone who makes a distribution
- Package maintainers
- System developers
- Documentation writers
- System Administrators

Unix Filesystems

- Have a hierarchical structure
- Employ consistent treatment of file data
- Have mechanisms for protection of file data

Categories of data

- Variable or non-variable (static)
- Shareability
  - host-specific
  - between hosts with same architecture and OS
  - between hosts with different architecture and same OS
  - between hosts with different flavor of OS
  - between hosts with different OS
  - between all hosts at a site
- Executables, special files, "data"
- etc.

You get a hierarchical structure

- "bin" and "sbin" for executables
- "etc" for site-local configuration
- "lib" for object libraries (arch-specific data)
- "libexec" for internal binaries (arch-specific data)
- "var" for all variable data
- /usr vs. root filesystem
- Additions for special cases: /tmp, /boot, /dev, /mnt, /home, etc.
- For something so heavily influenced by history, it's not that bad.

Major themes of new release

- add features for commercial systems
- multiple-architecture support
- clean up and clarify filesystem
- bring us closer to GNU and BSD where it helps improve things
- Also has less restrictive copying terms

Major changes since FSSTND

- /opt
- /usr/share
- /usr/libexec
- /var/cache
- /var/state
- /var/spool/mail moved to /var/mail
- OS independent base standard
- Linux specific annex

/opt

- For add-on application software packages that need a "playground"
- De-facto standard location
- Packages go into /opt/<package>
- Reserved for local control: /opt/bin, /opt/doc, /opt/include, /opt/info, /opt/lib, and /opt/man
- Host-specific configuration: /etc/opt
- Variable data: /var/opt

/usr/share

- For non-variable architecture independent data
- Intended to be shareable among all architecture platforms of a given OS
- Not OS independent data
- Used by BSD and GNU too
- New manual page structure (based on Linux plus BSD ideas) /usr/share/man/<locale>/man<section>/<arch>

/usr/libexec

- Internal binaries ("library executables")
- For executable programs run by other programs rather than by users
- internet daemons, getty, cc1, etc.
- Helps clean up /usr/lib

/var/cache

- Cached data via the filesystem
- Locally-generated as the result of time-consuming I/O or computation
- Can throw away the contents, therefore the data must be regeneratable
- Includes web proxies, formatted manual pages, and dynamically generated fonts
- Could someday include object files from compilations and other stuff

/var/state

- For per-application state data.
- /var/lib is deprecated
- Probably the most gratuitous change.

A few random topics

- read-only /usr hierarchy
- small root?
- what is static, what is variable?
- home directories and /root
- /boot

The Future

- Free conformance test suite
- More supplementary documentation
  - hier.7
  - implemention notes and patches
- Supplementary drafts
- Next generation of FHS?

That's all

- For more information: http://www.pathname.com/fhs/
- Any questions?