HP-UX 10.0 File System Layout

Whitepaper

Version 2.3

Last Modification: March 23, 1995

1. Introduction

1.1 Abstract

With the release of HP-UX 10.0, Hewlett-Packard will be introducing many new features for the HP9000 UNIX operating system. One of these is a new file system layout paradigm, modeled after AT&T SVR4 and OSF/1. The model provides many benefits such as separating the operating system from applications, providing a solid foundation for diskless and client/server file sharing models, and aligning HP with an industry-accepted file system layout. There are two main areas affected by the new file system layout: file/directory locations and system startup/shutdown control. As easy as this sounds, all software developers will be faced with design decisions when moving to the new model. The file system layout is a core technology upon which the operating system is built. Applications running on HP-UX 10.0 may need to make modifications to their products that enable them to correctly function within the new file system model.

Although the file system paradigm is modeled after AT&T SVR4 and OSF/1, HP is not implementing either of these operating systems with release 10.0. Rather, HP has implemented the file system model within HP-UX.

1.2 Purpose of Document

This document is intended to explain the HP-UX 10.0 file system layout, along with its benefits and impacts to software developers and users. After reading this document, a person should be able to:

- ☞ articulate the reasoning for implementing the new file system layout.
- ☞ assess the impacts of file system layout to products and/or user environments.
- ☞ design, build, and test the changes necessary to comply with the new file system layout.

1.3 Audience

This document is intended for HP-UX software developers, system administrators, site system planners, and users, both internal and external to Hewlett-Packard.

1.4 Identification

Within the computer industry, this file system layout is referenced by many different names: “AT&T SVR4 File System”, “V.4 File System”, “OSF/1 File System”, and “V4FS”. For the purposes of this document, the product will be referred to as “HP-UX 10.0 File System Layout” and “10FSL”. All references in this paragraph refer to the same topic and may be used interchangeably. However, implementations among vendors will differ.

1.5 Terms and Definitions

Please refer to the Glossary section (at the end of this paper) for terms and definitions used throughout this document.

1.6 Related Documentation

The following is a list of references used to design the HP-UX 10FSL. The reader is encouraged to consult them for more information:

[1] *filsys(BA_ENV)* man page
System V Interface Definition, Third edition, Volume 1
Page 5-40, UNIX Press, UNIX System Laboratories, Inc, 1992

[2] *hier(5)* manpage
OSF/1 Release 1.0 Programmer's Reference Manual
Page 3-6, Open Software Foundation, Inc., Nov 1990

[3] *hier*(5) manpage
HP-UX Reference, Volume 3, Release 9.0
Pages 816-819, Hewlett-Packard Company, August 1992

2. The New File System Layout

This chapter explains the 10.0 File System Layout (10FSL) and outlines the differences between HP-UX Releases 9.x and 10.0. All of the changes apply to both S700 and S800 computer systems. These differences will affect software developers, system administrators, and end users of HP-UX. For users of HP-UX, the changes will be seen primarily as new locations of many files and directories. For administrators, more changes will be apparent, including new configuration files and a new paradigm for system startup and shutdown scripts. For software developers, application behavior may need to be restructured to conform to the new layout.

These topics are covered in the following sections:

- **Introduction:** the rationale for, and an overview of, the layout.
- **File System Layout Specification:** the specification of the new layout.
- **Applications:** file system specifications for applications (non-OS products).
- **General Impacts to Developers and Users**: explanation of affected areas.
- **Solutions to Commonly Asked Questions:** task-oriented solutions to common questions.

2.1 Introduction

2.1.1 Philosophy of the New Layout

The 10.0 file system layout is similar to the OSF/1 and SVR4 layout. Important features of the HP-UX 10.0 file system layout include:

- The overall layout is based on a de facto industry standard. Customers with inter-vendor networks will find the HP-UX file system to be similar to that used by other vendors. When supporting multiple platforms, developers will require fewer HP-UX specific designs.
- Files are organized by various categories, such as static vs. dynamic, executable vs. configuration data, etc. This provides a logical structure to the file system, as well as providing the benefits listed in the following items.
- The operating system is kept separate from the applications on the system, and applications are kept separate from each other. This facilitates a manageable client/server file sharing model.
- Files that may be shared among hosts are kept separate from files that pertain only to a particular host. This facilitates sharing file system hierarchies over a network for both the diskless and client/server models.
- Configuration data, which is host-specific, is kept separate from the executable code that uses that data, allowing the code to be shared among hosts. In addition, since configuration data never appears in the same file as the code that uses it, changes made by the customer to the configuration data are not lost when installing future HP-UX updates.
- Important system configuration data is kept separate from temporary files, log files, and other by-products of system execution that are not required for correct operation of the system. This eases the tasks of system backups and disk space administration.
- The portion of the file system that is allowed to “grow” is restricted to a manageable subset of directories, simplifying the task of disk space administration.

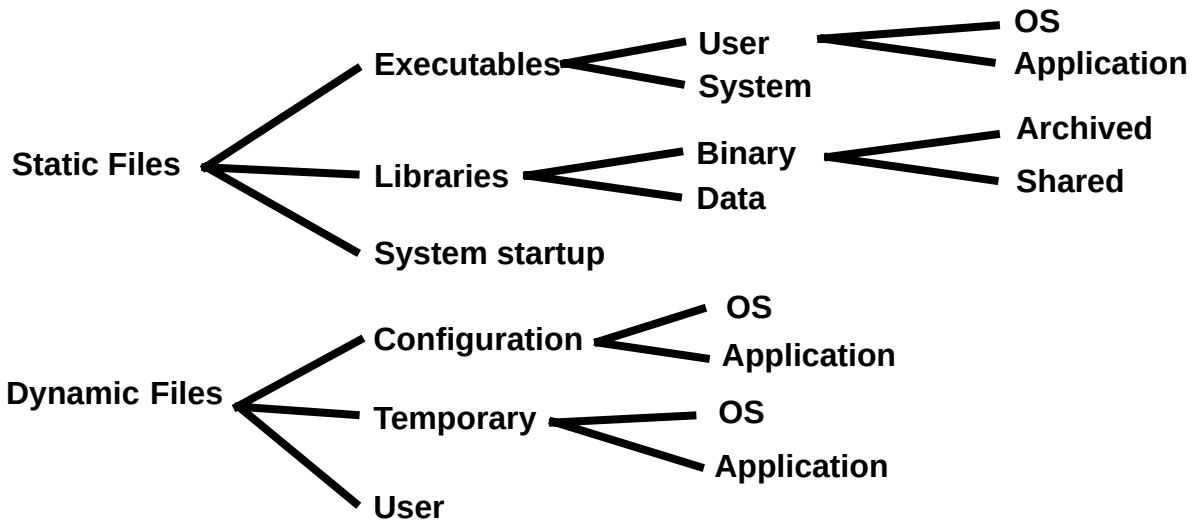
Although the 10.0 file system layout is based on the OSF/1 and V.4 file system layouts, it is not a “pure” implementation of either of these layouts. Industry compatibility is of great importance. However, it is

important to note that all of the major vendors supporting V.4 have slightly different implementations and their respective documentation should be consulted.

2.1.2 File System Layout Overview

One of the primary benefits of the 10.0 file system layout is that it categorizes and groups together files by functionality (static, dynamic, executable, configuration information, etc.). This organization is depicted in Figure 1.

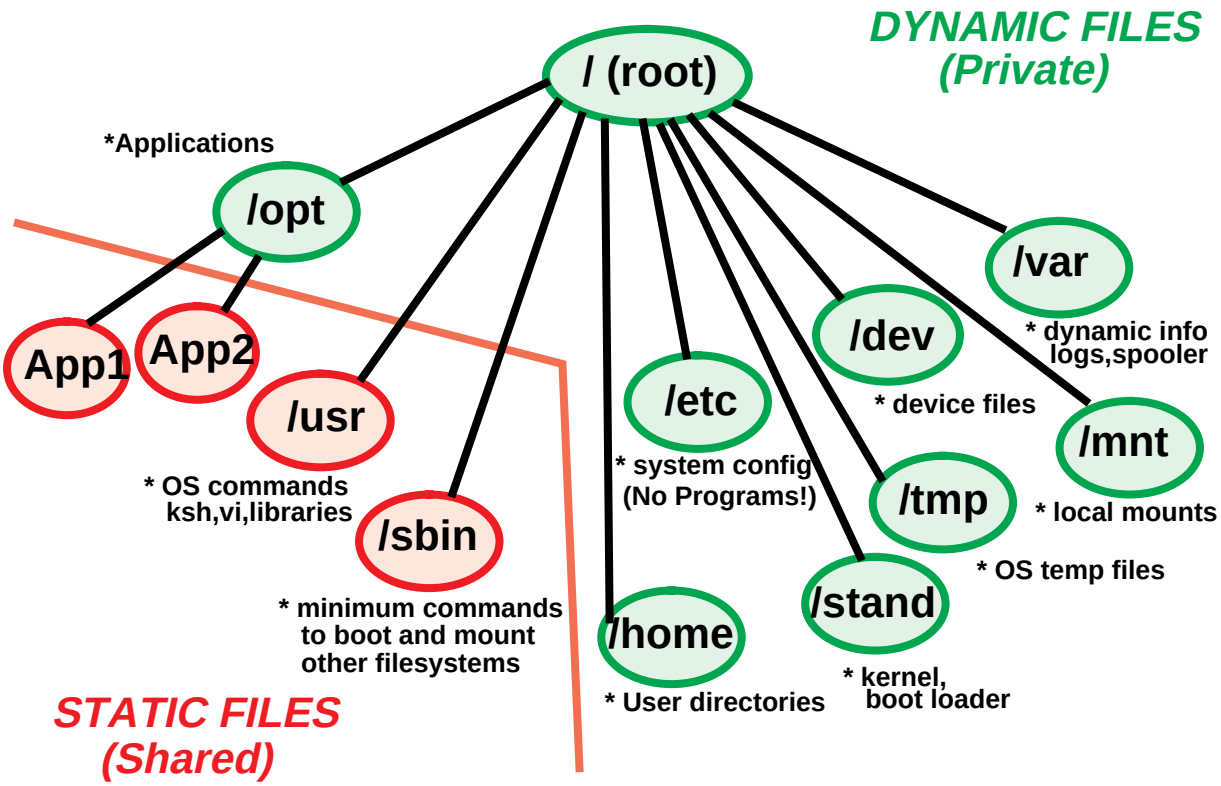
FIGURE 1. Generic File System Model



In the above figure, files are primarily categorized as *static* and *dynamic*. Dynamic files, which include files like log and configuration files, are further categorized by functionality (*configuration*, *temporary* & *user*). A similar division is seen with the static files, where files are categorized by three functionalities: *executable*, *library* and *system startup*. File types can further subdivided into *user* or *system*, *OS* or *Application*, etc., as illustrated in the figure above.

This grouping of files also supports the diskless paradigm by arranging the file system in a manageable fashion and allowing a clean sharing of file system resources in a client/server environment. In the diskless paradigm, directories are considered shared among clients (such as executables) or private to a client (e.g., configuration and temporary files). This shared vs. private distinction conforms well to the static vs. dynamic distinction given above; in 10.0, static files are shared, while dynamic files are private. Figure 2 represents the directory tree of the 10.0 file system layout, showing this private/dynamic vs. share/static distinction.

FIGURE 2. HP-UX 10.0 File System Layout



2.2 File System Layout Specification

This section describes the differences in the layout between HP-UX Release 9.x and 10.0.

2.2.1 General Definitions

The following are general rules for the new layout:

- The shareable portion of the OS resides beneath **/usr** and **/sbin**. Only the operating system may install files into these hierarchies.
- Applications reside in subdirectories beneath **/opt**. This is the recommended install point for applications.
- Directories **/usr**, **/sbin**, and the application subdirectories of **/opt** are shareable among networked hosts. These hierarchies must *not* contain host-specific information. All host-specific configuration data, temporary files, log files, and other files inappropriate for sharing among hosts must reside in private directories on the file system. The private directories include **/etc, /var, /tmp, /stand, /home** and others in Figure 2.
- The **/etc** directory is used exclusively for host-specific configuration data essential to the correct operation of the system. This directory no longer holds executable commands.

- The **/var** hierarchy contains host-specific files created during execution of the system that are not essential configuration information. Examples of files that live here are log files, temporary files, and printer spool files.
- The **/home** directory is the root for all users' home directories.
- The **/export** directory is the root for sharable, networked file systems, such as NFS exported directories.

2.2.2 10.0 Directory Definitions

The following table outlines most of the directories found in the HP-UX file system. For each directory, it shows the old directory name, where applicable, and the associated use for the directory. The last column of the table indicates whether the directory is shared among members of a cluster or is private to the host. Each directory is explained in detail in Section 2.2.3, "Directory Details".

Table 1: HP-UX 10.0 Directory Definitions

HP-UX 10.0	Pre-HP-UX 10.0	Description/Comments	Private/Shared
/dev	No Change	Device files for local devices	private
/etc	No Change	Machine-specific configuration and administration databases. No executables invoked by users.	private
/etc/opt/<application>	N/A	Application-specific configuration files.	private
/etc/rc.config.d	N/A	Startup configuration files.	private
/export	N/A	Default root of exported file systems	server directory
/export/private_roots	N/A	For host-specific files	server directory
/export/shared_roots	N/A	For shared OS and applications	server directory
/home	/users	Default for user directories	private
/home/<username>	/users/<username>	User home directory	private directory or local mountpoint
/lost+found	No Change	Storage directory for fsck	private
/mnt	No Change	Mounting point for local file systems	private
/net	No Change	Mounting point for remote file systems	private
/opt	N/A	Root for optional applications.	private
/opt/<application>	/usr/<application>	Application executables, libraries, and support files.	shared
/sbin	N/A	Essential system commands. (those needed to boot system and mount file systems)	shared

Table 1: HP-UX 10.0 Directory Definitions

HP-UX 10.0	Pre-HP-UX 10.0	Description/Comments	Private/Shared
/sbin/init.d	N/A	Startup and shutdown scripts	shared
/sbin/rc#.d	N/A	Startup and shutdown link files for script sequencing.	shared
/stand	N/A	Standalone machine-dependent binaries and kernel configs	private
/tmp	No Change	System-generated temporary files	private
/usr	No Change	Mount point for sharable user commands, libraries, and documentation.	shared
/usr/bin	/usr/bin and /bin	OS user commands	shared
/usr/ccs	N/A	Unbundled development package	shared
/usr/ccs/bin	N/A	Development binaries	shared
/usr/ccs/lib	N/A	Development libraries	shared
/usr/conf	/etc/conf	Kernel configuration.	shared
/usr/contrib	No Change	Contributed software.	shared
/usr/include	No Change	Header files.	shared
/usr/lbin	N/A	Backends to other commands.	shared
/usr/lib	/usr/lib and /lib	Object code and object code libraries.	shared
/usr/local	No Change	User contributed software	shared
/usr/newconfig	/etc/newconfig	Default operating system configuration data files.	shared
/usr/old	N/A	Obsolete files.	shared
/usr/sbin	N/A	System administration commands	shared
/usr/share	N/A	Architecture independent sharable files.	shared
/usr/share/dict	/usr/lib/spell	Dictionaries for **spell** and **ispell**	shared
/usr/share/lib	N/A	Misc sharable files.	shared
/usr/share/man	/usr/man	OS manpages.	shared
/var	N/A	Holds files created at runtime, such as log files and temporary files.	private
/var/adm	/usr/adm	Common administrative files and log files.	private

Table 1: HP-UX 10.0 Directory Definitions

HP-UX 10.0	Pre-HP-UX 10.0	Description/Comments	Private/Shared
/var/adm/crash	/tmp	Kernel crash dumps	private
/var/adm/cron	/usr/lib/cron	Cron queueing	private
/var/adm/sw	N/A	SD directory	private
/var/adm/syslog	N/A	Files generated by **syslog**	private
/var/mail	/usr/mail	Incoming mail	private
/var/news	/usr/news	News	private
/var/opt/<application>	N/A	Application-specific temporary or data files.	private
/var/preserve	/usr/preserve	Preserved editor files.	private
/var/run	N/A	PID files.	private
/var/spool	/usr/spool	Spooled files.	private
/var/spool/cron	/usr/spool/cron	Crontabs and at jobs.	private
/var/spool/locks	/usr/spool/locks	UUCP Lock files.	private
/var/spool/lp	/usr/spool/lp	Printer spooling.	private
/var/spool/mqueue	/usr/spool/mqueue	Outgoing mail.	private
/var/spool/sw	N/A	Default location for SD depot.	private
/var/spool/uucp	/usr/spool/uucp	UUCP spool directory	private
/var/spool/uucppublic	/usr/spool/uucppublic	Incoming UUCP files	private
/var/tmp	/usr/tmp	Application generated temporary files.	private
/var/uucp	/usr/spool/uucp	UUCP administration files	private

2.2.3 Directory Details

This section provides details on each of the directories listed in the previous two tables, both the 10.0 directories and the transition symbolic links.

2.2.3.1 /dev

/dev is used for device files. The contents and meaning of **/dev** is not changing. Nothing should be installed in **/dev**. Instead, configuration files should create node-specific device files.

2.2.3.2 /etc

The **/etc** hierarchy contains host-specific system and application configuration files important to the correct operation of the system. Files located here are usually fixed size and do not grow. By contrast, the **/var**

hierarchy holds files that are dynamic in length or are less critical to system execution, such as log files. In general, **/etc** holds essential information that must be preserved in order for the system to function correctly, while **/var** holds information generated by the system that may be disposed of when it is no longer of interest. Some customer sites may choose to make automatic backups of the **/etc** hierarchy, but not of **/var**.

There will be subdirectories under **/etc** for some systems; for example **/etc/mail** and **/etc/uucp** for the mail system and uucp respectively.

The **/etc** directory itself is used ONLY for system configuration files. **/etc** no longer contains commands, rc scripts, log files (with the exception of a few critical log files), or other files not related to system configuration. Most commands have moved to **/usr/sbin**. Rc scripts now reside in **/sbin/init.d** and follow the new system startup and shutdown paradigm (see Section 3., "System Startup/Shutdown Model"). Log files and other miscellaneous files not related to system configuration are now in **/var**.

2.2.3.2.1 /etc/opt

Applications will store application-specific, host-specific configuration data under **/etc/opt/<application>**. See section 2.3, "Applications", for more details.

2.2.3.2.2 /etc/rc.config.d

This directory contains configuration data files for startup and shutdown scripts.

2.2.3.3 /export

The directory is used to support diskless file sharing. Servers export root directory hierarchies for networked clients.

2.2.3.4 /home

User directories will be created and managed under **/home** instead of **/users**. Nothing should be installed in **/home**. This is a portion of the file system that is allowed to grow.

2.2.3.5 /lost+found

/lost+found contains files located by *fsck(1m)***.**

2.2.3.6 /mnt

Reserved name for mount points for local file systems. This is not an install location (i.e. nothing should be directly installed here from HP update media). **/mnt** can be used as a mount point directory, or a directory containing multiple mount points.

2.2.3.7 /net

Reserved name for mount points for remote file systems.

2.2.3.8 /opt

The root directory for optional applications. **/opt** is not a sharing point, but rather, its subdirectories are sharing points. This allows one to mount only those applications that make sense for a given machine or release, rather than mounting all applications available on a server. No files should be delivered into **/opt**, but rather into subdirectories of **/opt**, as described below.

2.2.3.8.1 /opt/<application>

The shareable portion of each optional application resides in a hierarchy beneath a single subdirectory of **/opt**. A uniform application structure is recommended, which consists of the following standard set of directories under **/opt/<application>**: **bin, lib, man, help, lbin** and **newconfig** for default configuration info. See section 2.3, "Applications", for more details.

2.2.3.9 /sbin

/sbin contains the commands and scripts essential to boot and shutdown a system. **/sbin** contains the commands required to bring the system into a state in which the **/usr** filesystem can be mounted and the boot process continued. **/sbin** also contains commands needed to fix filesystem mounting problems.

Commands in **/sbin** must not depend on any filesystems that may not be mounted at the time the command must execute, including the **/usr**, **/var**, and **/opt** filesystems. Since shared libraries reside beneath **/usr**, commands in **/sbin** are statically linked (i.e., built with archived libraries). Commands in **/sbin** must not execute any commands from the **/usr** filesystem; only other **/sbin** executables may be referenced. If a command referenced by an **/sbin** command is replicated between **/sbin** and either **/usr/bin** or **/usr/sbin**, then the path to the command must refer to the **/sbin** version.

Some commands in **/sbin** are duplicates of, or the target of symbolic links from, other commands in **/usr/bin** and /**usr/sbin**. For example, the *ls* command exists in both **/sbin** and **/usr/bin**. If a command in **/sbin** is duplicated in either **/usr/bin** or **/usr/sbin**, the duplicate exists to take advantage of shared libraries for most executions of that command, or to provide full functionality if the **/sbin** version does not.

Some commands in **/sbin** do not offer the full functionality provided by their **/usr/bin** or **/usr/sbin** counterparts. For example, some NLS functionality that requires shared libraries is not present in the **/sbin** versions of certain commands. For this reason, use of the **/usr/bin** or **/usr/sbin** versions of replicated commands is preferred, when possible. When constructing shell PATH variables that contain a **/sbin** component, **/sbin** should appear after **/usr/bin** and/or **/usr/sbin** in the path.

2.2.3.9.1 /sbin/init.d and /sbin/rc#.d

/sbin/init.d contains all rc scripts used to startup and shutdown various subsystems.

/sbin/rc#.d contains ordered symbolic links to rc scripts in **/sbin/init.d** that are executed when changing run levels.

See Section 3., "System Startup/Shutdown Model", for more information.

2.2.3.10 /stand

/stand is for system-specific kernel configuration and binary files. The files are typically needed at boot time to bring up a system. This directory is not an install point.

2.2.3.11 /tmp

/tmp is for system-generated temporary files. The contents of **/tmp** are usually NOT preserved across a system reboot. The choice of whether or not **/tmp** is cleaned up at boot time is left to the customer.

The **/tmp** directory is private. Since many sites will delete files from **/tmp** at boot time, files that must be preserved should not be placed in the **/tmp** directory. Application working files should go in **/var/tmp** or **/var/opt/<application>**. Files generated by the OS that must be preserved across reboots should go into the **/var/tmp** directory.

2.2.3.12 /usr

/usr contains the bulk of the operating system, including commands, libraries and documentation. The **/usr** file system contains only shareable operating system files, such as executables and ASCII documentation. Multiple systems of compatible architectures should be able to access the same **/usr** directories. **/usr** may be mounted as read-only by diskless clients, and thus may not be writable by clients.

The allowed subdirectories in **/usr** are defined below; no additional subdirectories should be created. Any 9.X applications that resided in other subdirectories in **/usr** have moved beneath the **/opt** hierarchy.

2.2.3.12.1 /usr/bin

/usr/bin is used for common utilities and applications. In general, commands documented in section 1 of the HP-UX Reference Manual reside in **/usr/bin**, while commands documented in section 1M reside in **/usr/sbin**.

2.2.3.12.2 /usr/ccs

The minimal C compiler is located here. The functionality is sufficient to build a kernel. The fully-functional C compiler resides below **/opt.**

2.2.3.12.3 /usr/conf

/usr/conf is a static directory containing the sharable kernel build environment.

2.2.3.12.4 /usr/contrib

This directory contains contributed software. The 10.0 layout has no changes to this directory.

2.2.3.12.5 /usr/include

/**usr/include** contains header files. The 10.0 layout has no changes to this directory.

2.2.3.12.6 /usr/lbin

The **/usr/lbin** directory is intended for backends to commands in the **/usr** hierarchy. Commands such as **/usr/lib/divpage** and **/usr/lib/diff3prog** are placed in **/usr/lbin.** There are some subdirectories for special systems, such as **/usr/lbin/spell** and **/usr/lbin/uucp**.

2.2.3.12.7 /usr/lib

/usr/lib holds libraries and machine dependent databases. In 10.0, most files that once resided in **/lib** now reside in **/usr/lib**. There is no **/lib** directory; code referencing **/lib** should be changed to reference the correct path.

2.2.3.12.8 /usr/local

/usr/local is for site local files, including binaries, libraries, sources, and documentation. HP will deliver this directory empty and not install software here.

2.2.3.12.9 /usr/newconfig

This directory contains default operating system configuration data files.

Files that once resided in /**etc/newconfig** might now reside in **/usr/newconfig** or **/opt/<application>/newconfig**. The structure of **/usr/newconfig** is different than that of **/etc/newconfig**: **/usr/newconfig** contains a directory hierarchy somewhat mirroring that of /.

2.2.3.12.10 /usr/old

During an operating system update, this directory is used for host customization. System files being replaced by files in **/usr/newconfig**, will be moved here. It is also used to hold old versions of software for compatibility with a previous release. **/usr/old** contains a directory hierarchy somewhat mirroring that of /.

2.2.3.12.11 /usr/sbin

The directory **/usr/sbin** is for system administration related commands. Many of the commands previously in **/etc** have moved to this directory. In general, commands documented in section 1M of the HP-UX Reference Manual are in **/usr/sbin**.

2.2.3.12.12 /usr/share

This hierarchy contains architecture-independent sharable files that can be shared among various architectures (e.g., terminfo files).

2.2.3.12.13 /usr/share/dict

This directory contains **spell** and **ispell** dictionaries.

2.2.3.12.14 /usr/share/doc

This directory contains HP-UX operating system documentation on various topics that is not delivered with other parts of the system.

2.2.3.12.15 /usr/share/lib

This directory is for miscellaneous sharable files. For example, **terminfo** files will appear beneath this directory.

2.2.3.12.16 /usr/share/man

This directory is for man pages. Processed man pages (e.g., **/usr/share/man/cat1.Z/***) will also be held here.

2.2.3.12.17 /usr/tmp

A temporary directory symbolically linked to **/var/tmp** for backward compatibility. This directory is *not shared* with other systems in a diskless cluster.

2.2.3.13 /var

This directory is for multipurpose log, temporary, transient, variable sized, and spool files. The **/var** directory is extremely “variable” in size, hence the name. In general, any files that an application or command creates at runtime, and that are not critical to the operation of the system, should be placed in a directory that resides under **/var**. For example, **/var/adm** will contain log files and other runtime-created files related to system administration. **/var** will also contain variable size files like crontabs, and print and mail spooling areas.

In general, files beneath **/var** are somewhat temporary. System administrators that wish to free up disk space are likely to search the **/var** hierarchy for files that can be purged. Some sites may choose not to make automatic backups of the **/var** directories. If a product locates important configuration files here that do not fit under **/etc**, it is recommended that documentation explicitly reference **/var** files to back-up.

/var should not be placed on a small, fixed-size partition. Also, **/var** is not an install point.

2.2.3.13.1 /var/adm

This directory hierarchy is used for common administrative files, logs, and databases. For example, files generated by *syslog*(3C), files used by *cron(1M)*, and kernel crash dumps will be kept here and in subdirectories. Host-specific administration information will also be kept here. **/usr/adm** has become **/var/adm.**

2.2.3.13.2 /var/adm/crash

Kernel crash dumps will be located in this directory.

2.2.3.13.3 /var/adm/cron

Used for log files maintained by cron, and cron fifos.

2.2.3.13.4 /var/adm/sw

Used by SD, the HP OpenView Software Distributor.

2.2.3.13.5 /var/adm/syslog

System log files generated by **syslog** (see *syslogd(1M)* and *syslog(3C)*) will go into this directory.

2.2.3.13.6 /var/mail

Directory where incoming mail messages are kept.

2.2.3.13.7 /var/news

Electronic bulletin board files used by *news(1)* will be kept here. Formerly **/usr/news**.

2.2.3.13.8 /var/opt

Application runtime files (e.g., logs, status, temporary files) for applications mounted in **/opt** will be stored in **/var/opt/<application>** for each application.

2.2.3.13.9 /var/preserve

Files preserved by *vi(1)* will be stored here. Formerly **/usr/preserve**.

2.2.3.13.10 /var/run

PID files for daemon programs will be stored in **/var/run**, NOT in **/etc**.

2.2.3.13.11 /var/spool

Host-specific spool files are located here. In general, **/usr/spool** becomes **/var/spool.**

2.2.3.13.12 /var/tmp

/var/tmp is for user temporary files generated by commands in the **/usr** hierarchy. Files located here are preserved between system reboots. Temporary files generated by applications installed under **/opt/<application>** will use **/var/opt/<application>** for temporary files.

2.2.3.13.13 /var/uucp

UUCP administration files will reside here.

2.3 Applications

As discussed earlier, the 10.0 file system model separates static and dynamic files. Application product file system layouts follow the same conventions and extend the model to provide separation among products. This is accomplished in many ways:

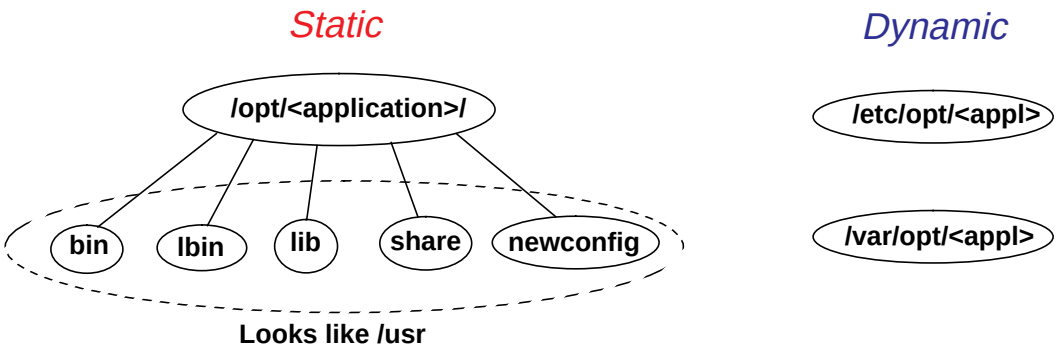
- In general, applications are located separate from the operating system under **/opt**. They do not reside below **/usr** and **/sbin**, which are OS directories. This allows for separate application product directories that can span OS revisions, and allows easier disk partitioning management.
- System administrators are able to easily identify the location of all files related to an application.
- Critical information is easily backed-up because all application configuration files are contained beneath one hierarchy.

2.3.1 File Layout

In general, **/opt** is the application directory. Below here, applications are self-contained in their own directory, **/opt/<application>**. Shareable files (that may be shared among multiple nodes), such as binaries, man pages and libraries, reside in this directory. Host-specific private files, such as logs and node-specific configurations, are located in the private/dynamic directories under **/var/opt/<application>** and **/etc/opt/<application>**, respectively.

The application root (**/opt/<application>**) should contain a subdirectory structure similar to that of **/usr**: **bin, lbin, lib, man and newconfig**. FIGURE 3., "Application File Layout", shows a sample directory structure.

FIGURE 3. Application File Layout



2.3.2 Commands and Libraries

Application binaries and libraries follow the same layout used by the operating system. User commands are located in **/opt/<application>/bin**, backend commands reside in **/opt/<application>/lbin**, and shared and archived libraries reside in **/opt/<application>/lib**.

2.3.2.1 NLS Message Catalogs

NLS message catalogs for applications should be placed in the **/opt/<application>/lib/nls** directory. To automatically access these files using the standard NLS interfaces, the NLSPATH environment variable must be set to include the application's message catalog directory. Applications are responsible, upon

invocation of any application binaries, to add the correct component to the NLSPATH environment variable.

2.3.3 Manpages and Other Architecture-Independent Files

Application manpages are kept separate from OS manpages and reside in **/opt/<application>/share/man**.

Other architecture-independent files that may be shared by more than one system reside below **/opt/<application>/share** and must follow the same layout conventions specified for the operating system in section 2.2, "File System Layout Specification". These include object files and libraries, to name a few.

2.3.4 Configuration Files

System-dependent configuration files for applications reside in **/etc/opt/<application>** and include files necessary for the proper runtime execution of the product. Files that are by-products of execution are considered temporary files and should be located under **/var/opt/<application>**. Because all system configurations are located under **/etc**, system administrators can easily view system and application configurations. This also facilitates backup of configuration data for a given machine.

NOTE:	Dynamic files, such as logs and configurations, should never be loaded directly into their target locations in **/etc** or **/var** (This may overwrite current data). These should be delivered below **/opt/<appl>/newconfig**, as specified in section 2.2.3.12.9, "/usr/newconfig", and moved into place during product configuration.

2.3.5 Logs and Temporary Files

Temporary files created by applications (i.e., products installed in **/opt**) for use during runtime are located in **/var/opt/<application>**. This includes logs and lock files that are by-products of runtime execution, but does not include those configuration files necessary for correct execution of the application. Configuration files reside below **/etc/opt.**

2.3.6 Path Variables

Product binaries, man pages and other files are no longer contained in common system directories. Consequently, the default system *path* values no longer point to all the installed software and must be adjusted for products as they are added to a system. The new model provides a mechanism to ‘register’ a product’s files with the appropriate *path* variables.

Two path variables are considered: PATH and MANPATH. Each variable is addressed with a separate file, each containing a colon-separated list of values representative of the particular *path* variable:

/etc/PATH	location of commands
/etc/MANPATH	location of man pages

The system default login files (**/etc/profile**, **/etc/csh.login**) appropriately source these files for the respective *path* variables.

The *path* files should never be directly manipulated by a product's installation processes. SD provides configuration utilities that enable applications to add their product-specific values to the appropriate *path* variable files.

2.4 General Impacts to Developers and Users

The 10FSL introduces many changes that affect not only software developers, but also users and system administrators. The major change involves filename and directory locations. If you have existing code that needs to be integrated into HP-UX 10.0, some of the common impacts are discussed below.

2.4.1 Embedded Pathnames

Software developers frequently code pathnames into their source. These paths may appear in header files and library routines, as well as the main body of code. The 10FSL changes require developers to scan these sources for pathname values, evaluate the findings and implement the necessary changes to reflect the correct file system layout. Hard-coded pathnames may be found in:

- ✓ Program source, headers and make files.
- ✓ Scripts (ksh, psh, sh, awk, sed, etc.)
- ✓ Libraries
- ✓ Binaries and Object code
- ✓ Message catalogs and help/man files

For solutions to finding embedded pathnames, see section 2.5.1, "Finding Embedded Pathnames."

2.4.2 Environment Variables

An important, but often overlooked, area is environment variables. These may be contained in the system and user login files such as ~/.profile, ~/.login, ~/.vueprofile, ~/.cshrc, and /etc/skel/d.*. These are the most common files, but there may be others associated with applications. Be sure to check your system for other possible occurrences.

Environment variables that have execution strings or contain paths are likely to be impacted. Be sure to change all path variables to reflect the correct directories where your commands are contained.

2.4.3 Build & Test Environments

Developers often keep separate environments for building and testing their systems and applications. This is also an important area to evaluate for hard-coded pathnames. Makefiles and test scripts may contain paths that have been changed as a result of the 10FSL.

2.4.4 Documentation

Many applications have detailed documentation outlining the product installation locations. Documentation should also be scanned for obsolete path names.

2.5 Solutions to Commonly Asked Questions

This section includes both tips and answers to commonly asked questions regarding the file system layout and the upgrade from HP-UX 9.x to 10.0.

2.5.1 Finding Embedded Pathnames.

There are a number of ways to search for embedded pathnames. A few recommended solutions are listed below. However, you should note that these are only suggestions and are not guaranteed to find all embedded pathnames. Each developer should analyze their applications and systems for any other areas that may not execute properly on the 10FSL.

2.5.1.1 HP Tools

Hewlett-Packard will provide tools to help users identify and fix absolute pathnames that are obsolete. The tools will convert shell scripts (ksh, sh, csh, etc.) and makefiles to comply with the new 10FSL. The same tools that operate on scripts and makefiles will also do simple string translations for absolute pathnames found in ordinary ASCII files. On a HP-UX 10.0 system, one such tool is **/opt/upgrade/bin/analyzer**. Please see its accompanying documentation for further details.

2.5.1.2 ‘strings’ command

The *strings* command is a good alternative. This is most effective on non-ASCII files, but may be used on ASCII text files. See the HP-UX manual page, *strings(1)*, for more details. A sample command is shown below:

```
strings -a <filename> | grep / | sort -u
```

This will produce a unique sorted list of alphanumeric strings containing ‘/’, indicating either an absolute or relative pathname. This will probably generate a lot of non-pathname results and will need to be searched for valid pathnames.

On a pre-10.0 system, the *strings* command may be found in **/usr/bin**. On a 10.0 system, *strings* is found under **/usr/ccs/bin.**

2.5.2 Finding New Locations of Files

HP will provide a database and a tool to assist you in locating operating system product files in both the 9.x and 10.0 locations. The tool is **/opt/upgrade/bin/fnlookup** on an HP-UX 10.0 system. Please see its accompanying documentation for further details.

In the meantime, you may examine a HP-UX 10.0 system using the *find* command:

```
find <path> -print | grep <filename>
```

This will produce the location of the desired file if it is contained below <path>. If not, try another path location.

2.5.3 Building PATH Variables

In the 10FSL, there are no longer only a few key places, such as **/usr/bin**, where executables may be found. Applications will have both manual (man) pages and binaries in application directories below **/opt**.

As applications are added to a user's environment, the PATH and MANPATH variables must be updated so that commands and man pages will be found. There is an automated mechanism available to application developers to add their appropriate entries to these paths. However, all applications may not take advantage of this mechanism. In this case, the user will need to manually add to the appropriate path variables.

For example, if a (ksh, sh) user has application XYZ and wishes to have the man pages and binaries accessible without fully qualifying the path, both the PATH and MANPATH variables may be updated to resemble:

```
PATH=$PATH:/opt/XYZ/bin
MANPATH=$MANPATH:/opt/XYZ/share/man
```

This task would be executed for each application that was added to the user's environment. In some cases, applications may provide this capability during configuration. In other cases, system administrators and users must ensure their variables are set correctly for the environment. This syntax is specific for ksh and sh; csh users must use the appropriate syntax.

3. System Startup/Shutdown Model

This chapter explains the 10.0 system startup and shutdown model and outlines the differences between HP-UX Releases 9.x and 10.0. All of the changes apply to both S700 and S800 computer systems. These differences will primarily affect software developers and system administrators. Software developers will construct rc files in a different manner, specifying run-level execution order and enabling control through configuration variables. System administrators will control subsystem behavior, on a per-host basis, by modifying variables that control subsystem startup and shutdown.

These topics are covered in the following sections:

- **Introduction:** the rationale for, and an overview of, the model.
- **Startup/Shutdown Specifications:** the specifications of the new model.
- **General Developer and User Impacts:** explanation of affected areas.

3.1 Introduction

3.1.1 Philosophy of the New Startup/Shutdown Model

The 10.0 Startup/Shutdown model is based upon the scheme used by the OSF/1 operating system. Important new features of the scheme include:

- The model separates the execution scripts from the configuration information required for execution. Within a single directory, system administrators may easily modify the behavior of the startup/shutdown sequence by changing configuration variables.
- Execution scripts are not modifiable. System administrators cannot change the scripts themselves, but rather they are provided with configuration variables that change the behavior of the scripts.
- Individual subsystems may now be started or stopped on a per-init-state basis. This enables fine levels of control and subsystem separation.

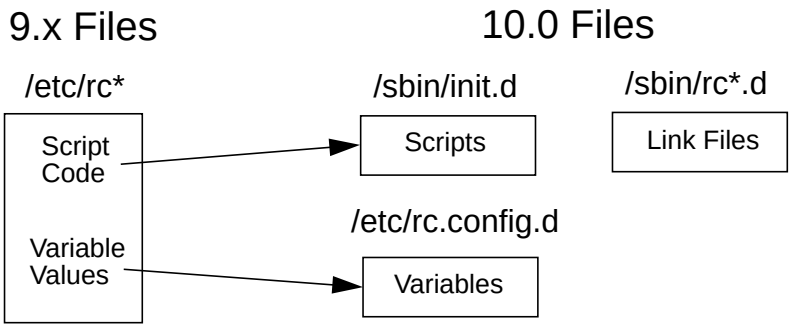
3.1.2 Startup/Shutdown Overview

The old **/etc/rc**-based startup scheme will be replaced by the startup and shutdown scheme used by the OSF/1 operating system. The new startup/shutdown scheme consists of three parts:

1. Execution Scripts used to startup and shutdown individual subsystems are contained in **/sbin/init.d**.
2. Configuration files for each execution script are contained in **/etc/rc.config.d** and reside in a filename that is generally equal to the script for which they configure.
3. Link Files in **/sbin/rc*.d** control the sequencing order of the execution scripts.

FIGURE 4., "9.X - 10.0 RC Configuration Mapping", shows the relationship of 9.x files to 10.0 files. Prior to 10.0, both script code and configuration information, such as IP addresses and hostnames, were contained within the files. The new model partitions the configuration data from the scripts. Administrators use the configuration variables in **/etc/rc.config.d** to change the behavior of scripts in **/sbin/init.d.** Script sequencing is also a new feature for 10.0 and is controlled through link files in **/sbin/rc*.d** directories.

FIGURE 4. 9.X - 10.0 RC Configuration Mapping



3.2 Startup/Shutdown Specifications

This section describes the new startup and shutdown model components: execution scripts, configuration variable scripts and link files.

3.2.1 Execution Scripts

The **/sbin/init.d** directory contains all scripts used to startup and shutdown various subsystems. No script may invoke any of the other scripts in this directory. Scripts obtain configuration data from variables in **/etc/rc.config.d** (more on this later), which must be sourced by the execution script. All files in the **/etc/rc.config.d** directory may be read by sourcing the single file **/etc/rc.config**.

NOTE:	The startup and shutdown scripts shipped by HP in **/sbin/init.d** must *not* be edited. Any changes made to these scripts will be overwritten when a new software release is installed. Modifying the behavior of a subsystem script is accomplished by using configuration variables, discussed in 2.3.4, "Configuration Files".

In general, each script under **/sbin/init.d** should perform both the startup and shutdown functions. In order to control the functionality within the script, each must also support standard arguments and exit codes. Scripts must be written for the POSIX shell. A template script may be found in **/sbin/init.d/template.**

3.2.1.1 Arguments to scripts

The startup/shutdown scripts must recognize the following four arguments:

- **start_msg.** The “start_msg” argument is passed to scripts so the script can report back a short message indicating what the “start” action will do. For instance, when the lp spooler script is invoked with a “start_msg” argument, it echoes “Starting the LP subsystem”. This string is used by the startup checklist. Note that when given just the “start_msg” argument, scripts will only print a message and NOT perform any other actions.
- **stop_msg**. The “stop_msg” argument is passed to scripts so that the script can report back a short message indicating what the “stop” action will do. For instance, when the lp spooler script is invoked with a “stop_msg” argument, it echoes “Stopping the LP subsystem”. This string is used by the shutdown checklist. Note that when given just the “stop_msg” argument, scripts will only print a message and NOT perform any other actions.

- **start**. Upon receiving the “start” argument, the script should start the subsystem. All output should be echoed to *stdout*.
- **stop**. Upon receiving the “stop” argument, the script should shutdown the subsystem. All output should be echoed to *stdout*.

The messages echoed by the execution script when **start_msg** and **stop_msg** arguments are passed should contain a single line message with no more than 30 characters.

When passed the **start** and **stop** arguments, an execution script must not echo any messages indicating the entry or exit from the script. **Stop_msg** and **start_msg** arguments will be passed to the execution scripts by **/sbin/rc** to record these messages both on screen and in log files, indicating the beginning and ending of the script.

3.2.1.2 Naming Conventions

The startup and shutdown scripts are named after the subsystem they control. For example, the **/sbin/init.d/cron** script controls the cron daemon.

3.2.1.3 Scripts and Console Output

To ensure proper reporting of startup events, startup scripts will be required to comply with a few guidelines for script output and exit values.

Status messages, such as “starting foobar daemon”, must be directed to *stdout*. All error messages must be directed to *stderr*. Both *stdout* and *stderr* are redirected to the log file **/etc/rc.log**, unless the startup checklist mode is set to the raw mode. In this case, both *stdout* and *stderr* output go to the console.

Startup scripts, and the daemons or binaries they execute, must not send messages directly to the console during system boot or shutdown. This restriction exists because console output during the boot or shutdown sequence will overwrite the graphical checklist, leaving the checklist unreadable. These messages should be directed to *stdout* or *stderr*.

3.2.1.4 Exit values

Exit values for startup scripts are as follows:

0 -- script exited without error. This causes the status “OK” to appear in the checklist.

1 -- script encountered errors. This causes the status “FAIL” to appear in the checklist.

2 -- script was skipped due to overriding control variables from **/etc/rc.config.d** files or for other reasons, and did not actually do anything. This causes the status “N/A” to appear in the checklist.

3 -- script executed normally and requires an immediate system reboot for the changes to take effect (NOTE: Reserved for key system components).

NOTE:	These are the only exit values acceptable. Returning an arbitrary non-zero exit value from a command to indicate failure may cause the script to appear to have been skipped in the checklist, or may cause the system to reboot!

3.2.2 Configuration Variable Scripts

Instead of spreading configuration data throughout the various rc files in the system, configuration data is structured as a directory of files that allows developers to create and manage their own configuration files, without the complications of shared file ownership. The directory that holds the configuration variable scripts is **/etc/rc.config.d**. The configuration variable scripts will be sourced by the startup and shutdown execution scripts in **/sbin/init.d** during system startup and shutdown. This configuration information is used by the execution scripts to enable/disable and configure subsystems (such as IP addresses).

Examples of these variables include:

- HOSTNAME: Internet name of your system.
- IP_ADDRESS[0]: Internet address of your system, in dot format.
- SUBNET_MASK[0]: Internet subnetmask.

Each execution script may source its variables in one of two ways. If the execution script only needs the variables delivered with its product or fileset, it may explicitly source the file in **/etc/rc.config.d.** If the script requires variables that are delivered by other products or filesets, it may just source **/etc/rc.config,** which is a script that sources all the files below **/etc/rc.config.d**.

NOTE:	There must is no requirements on the order of the files sourced. This means configuration files must not refer to variables defined in other configuration files, since there is no guarantee that the variable being referenced is currently defined.

Configuration variable scripts are written for the POSIX *sh* (**/usr/bin/sh** or **/sbin/sh**), and not the Bourne *sh, ksh,* or *csh*. In some cases, these files must also be read, and possibly modified by other scripts or the SAM program. For this reason, each variable definition must appear on a separate line, in the syntax:

variable=*value*

No trailing comments may appear on a variable definition line. Comment statements must be on separate lines, with the “#” comment character in column 1. An example of the required syntax for configuration files is given below.

```
# Cron configuration. See cron(1m)
#
# CRON: Set to 1 to start cron daemon
#
CRON=1
```

The name of a configuration script in **/etc/rc.config.d** corresponds to the names of the associated startup and shutdown scripts found in **/sbin/init.d**.

3.2.2.1 /etc/TIMEZONE

/etc/TIMEZONE contains the definition of the TZ environment variable. It is sourced by **/etc/rc**.**config** along with the other **/etc/rc.config.d/*** files.

3.2.3 Sequencing Scripts with Link Files

The third aspect of the startup/shutdown scheme is controlling the order of script execution with link files. An important feature enables developers to control individual subsystems among run-levels. This is accomplished by providing link files in run level directories that point to the scripts in **/sbin/init.d**.

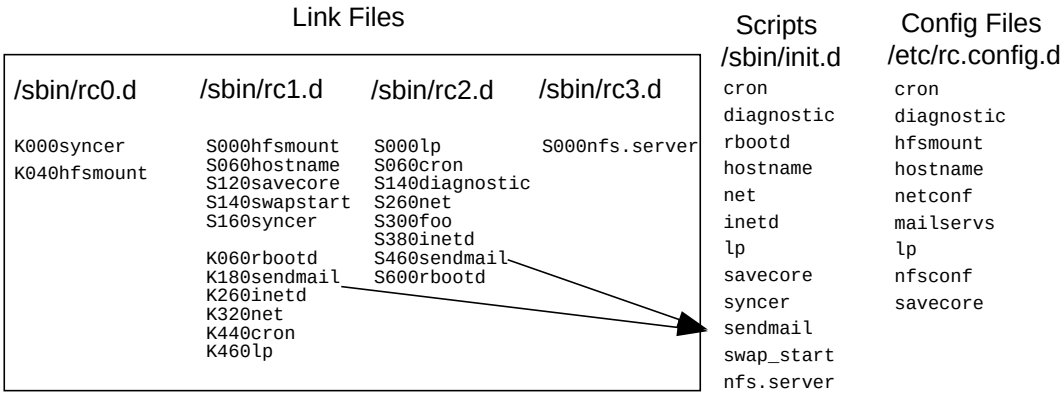
3.2.3.1 Run-level Directories: /sbin/rc#.d

The **/sbin/rc#.d** (where # is a run-level) directories are startup and shutdown sequencer directories. They contain only symbolic links to startup/shutdown scripts in **/sbin/init.d** that are executed by **/sbin/rc** on transition to a specific run level. For example, the **/sbin/rc3.d** directory contains symlinks to scripts that are executed when entering run level 3. (There is more information on **/sbin/rc** in Section 3.2.4, "Run Levels and /sbin/rc").

These directories contain two types of link files: *start* links and *kill* links. Start links have names beginning with the capital letter “S” and are invoked with the “start” argument at system boot time or on transition to a higher run level. Kill links have names beginning with the capital letter “K” and are invoked with the “stop” argument at system shutdown time, or when moving to a lower run level.

Further, all link files in a sequencer directory are numbered to ensure a particular execution sequence. Each script has, as part of its name, a three digit sequence number. This, in combination with the start and kill notation, provides all the information necessary to properly startup and shutdown a system**.**

FIGURE 5. Startup/Shutdown Component Relationships.



Note: The sequence numbers above are only for example and may not accurately represent your system.

Figure 5 shows the run-level directories and the relationship of the link files to the scripts. Because each script in **/sbin/init.d** performs both the startup and shutdown functions, each will have two links pointing towards the script from **/sbin/rc*.d**: one for the start action and one for the stop action.

3.2.3.2 Naming Conventions

The naming conventions for the link files are as follows:

/sbin/rc2.d/S060cron
1 2 3 4

The various components have the following meanings:

1. Run Level Number: The sequencer directory is numbered to reflect the run-level for which its contents will be executed. In this case, Start scripts in this directory will be executed upon entering run-level 2 from run-level 1, and Kill scripts will be executed upon entering run-level 2 from run-level 3.
2. Sequencing Type: The first character of a sequencer link name determines whether the script is executed as a start script (if the character is “S”), or as a kill script (if the character is “K”).
3. Sequence Number: A three digit number is used for sequencing scripts within the sequencer directory. Scripts are executed by type (start or kill) in lexicographical order.
4. Script Name: Following the sequence number is the name of the startup script. This name must be the same name as the script to which this sequencer entry is linked. In this example, the link points to **/sbin/init.d/cron.**

Scripts are executed in lexicographical order. The entire file name of the link is used for ordering purposes. When adding new sequencer entries, sequencer numbers are chosen to allow for gaps so that future entries may be inserted without requiring renumbering of existing entries. HP-supplied sequencer entries will all have unique numbers. (There is more information on sequence numbers in Section 3.2.5, "Link File Sequence Number Rationale and Assignment").

Subsystems are killed in the opposite order they were started. This implies that kill scripts will generally not have the same numbers as their start script counterparts. For example, if two subsystems must be started in a given order due to dependencies (e.g., **S111fubar** followed by **S222uses_fubar**), the counterparts to these scripts must be numbered so that the subsystems are stopped in the opposite order in which they were started (e.g.**, K555uses_fubar** followed by **K777fubar**).

Also, kill scripts for start scripts in directory **/sbin/rc**N**.d** reside in **/sbin/rc**(N-1)**.d**. For example **/sbin/rc3.d/S123foobar** and **/sbin/rc2.d/K654foobar** might be start/kill counterparts.

NOTE:	HP-UX will continue to support short filenames (i.e. 14 characters). Because of the filename place holders for ‘K’, ‘S’ and the 3 digit sequence numbers, subsystem script names in **/etc/init.d** are restricted to 10 characters in length.

3.2.4 Run Levels and /sbin/rc

In previous HP-UX releases, **/etc/rc** was run only once. Now it may run several times during the execution of a system, sequencing the execution scripts when transitioning run levels. However, only the subsystems configured for execution, through configuration variables in **/etc/rc.config.d**, are started or stopped when transitioning the run levels.

/sbin/rc sequences the startup and shutdown scripts in the appropriate sequencer directories in lexicographical order. Upon transition from a lower to a higher run level, the start scripts for the new run level and all intermediate levels between the old and new level are executed. Upon transition from a higher to a lower run level, the kill scripts for the new run level and all intermediate levels between the old and new level are executed.

When a system is booted to a particular run level, it will execute startup scripts for all run levels up to and including the specified level (except run level 0). For example, if booting to run level 4, **/sbin/rc** looks at the old run level (S) and the new run level (4) and executes all start scripts in states 1, 2, 3, and 4. Within each level, the start scripts are sorted lexicographically and executed in that order. Each level is sorted and executed separately to ensure that the lower level subsystems are started before the higher level subsystems.

Consequently, when shutting down a system, the reverse takes place. The kill scripts are executed in lexicographical order starting at the highest run level and working down, as to stop the subsystems in the reverse order they were started. As mentioned earlier, the numbering is reversed from the startup order.

States 0 and S are special cases. When entering state 0 or S, **/sbin/rc** will run start scripts in **/sbin/rc0.d**. Start scripts in state 0 are quick system administration scripts that prepare the system for a shutdown. When entering state S, all processes are killed and the system is in single user state. When entering state 0, the system is halted. The table below summarizes the run level definitions.

Table 2: Run Level Definitions

Run level	State	Sequencer Dir	/sbin/rc interaction
0	Halted	/sbin/rc0.d	All start and kill scripts executed
S	Single User	/sbin/rc0.d	
1	Minimal System Configuration	/sbin/rc1.d	When entering from lower state, all start scripts are executed. When entering from higher state, all kill scripts are executed
2	Multi-User	/sbin/rc2.d	
3	Exported File Systems	/sbin/rc3.d	
4	HP-VUE	/sbin/rc4.d	
5,6	Not currently used.	/sbin/rc5.d /sbin/rc6.d	

3.2.5 Link File Sequence Number Rationale and Assignment

To enable proper system startup and shutdown, execution scripts must be invoked in the correct order. This is accomplished by assigning sequence numbers for the link files (e.g., S300cron) contained in the sequencer directories (i.e., **/sbin/rc2.d**). The HP-UX 10.0 operating system and its associated products have been structured using a paradigm that groups various related functions into the same run state. This is accomplished by reserving blocks of sequence numbers for these functions. The paradigm is described in the sections below. Some entries provide examples of subsystems that are started at a particular level. Please consult an HP-UX 10.0 system for a complete and accurate listing.

3.2.5.1 Run Level 1 Paradigm

Run level 1 provides essential services, such as mounting file systems and configuring essential system parameters.

0XX	reserved for temporary links
1XX	mount local filesystems
2XX	essential process initialization/kill
3XX	set essential system parameters (hostname, lan address)
4XX	set other system parameters (date, privilege groups)
5XX	start essential daemons (swapper and syncer daemons)
6XX-8XX:	not currently used
9XX	reserved for future expansion

3.2.5.2 Run Level 2 Paradigm

Run level 2 is the general multi-user run state where most services are started.

0XX	reserved for temporary links
1XX	software installation/configuration (SD)
2XX	essential local daemons and services, started before network startup (clean log/tmp files, syslogd)
3XX	network startup
30X	network tracing/logging must be first
31X-33X	network low-level services (FDDI,ATM,Fiber,token ring)
34X	traditional TCP/IP initialization (ifconfig,route,gateway,netmask,etc.)
35X-39X	other network startup (x25, loopback daemon, naming daemon)
4XX	NFS/NIS initialization
5XX-6XX	services built on top of network services (DCE,DFS,NCS, rbootd, NetLS, mail) (Also client/server services: X font server, Kanji server)
500	inetd super-server
7XX-8XX	other local daemons/services (lp, cron, diagnostics, auditing, accounting, etc.)
9XX	reserved for future expansion
900	“Don’t Care” number for run state 2

3.2.5.3 Run Level 3 Paradigm

Run level 3 is the networked multi-user state. This run level is used to export file systems. Currently HP only supports NFS exports. Other vendors (Solaris) also do RFA exports here.

0XX	reserved for temporary links
1XX	NFS exports (NFS server)
2XX-8XX	not currently used
9XX	reserved for future expansion

3.2.5.3.1 Run Level 4 Paradigm

Run level 4 is reserved for desktop managers. Currently, HP-VUE is started in this run level, but as an entry in **/etc/inittab**. No start/kill links are currently shipped in run level 4.

3.2.6 Adding Your Own Subsystems

System administrators and software product developers may have a need to add their subsystem(s) to this model. This may be easily accomplished by following a few steps.

A good place to start is with the execution script. This may be derived from any of the system execution scripts found in **/sbin/init.d.** There is also a blank template in **/sbin/init.d/template**. The guidelines for execution scripts must be strictly followed such that the interface with **/sbin/rc** is correctly maintained. Any failure to do so will likely cause the execution script to fail.

A configuration file should also be added to **/etc/rc.config.d**. This file should contain a variable assignment that enables the user or system administrator to enable or disable the particular subsystem. The file should also contain any other variable assignments required by the subsystem during startup. The configuration file should not contain any executable script code other than variable assignments.

Selecting a run state and a sequence number should probably be done last. Many times it is difficult to determine exactly where the subsystem should fit until it has been coded and tested.

3.2.6.1 Selecting Sequence Numbers

All of the sequence numbers for HP products have been carefully assigned to ensure correct ordering, eliminate overlap and allow room for growth. Each of these products has registered with a central organization and has been given specific sequence numbers based on a number of factors about the product. When choosing sequence numbers for products or applications, system administrators and software product developers should adhere to the guidelines below.

Commercially available products may require specific sequence number ordering. Software developers should not arbitrarily choose an unused/unassigned number that happens to be absent from the system or table they are reviewing. Absent numbers that appear unused may have already been assigned by Hewlett-Packard to a product that has not yet been released, or is not installed on the system. Instead, developers should contact HP through:

> PA-RISC Developers Program
> (508) 436-5144
> pard@apollo.hp.com
> Ask for information regarding “Startup/Shutdown Sequence Number Assignments”

Developers will be asked a few questions about the product and assigned a set of numbers (start and kill). The product will then be registered with Hewlett-Packard and be ensured that no other registered products will have duplicate numbers that may corrupt an end-user’s system. However, duplicate numbers may be assigned for applications that do not conflict. Developers are encouraged to contact HP late in their develoopement cycle, as close to application release as possible.

If specific ordering is not a concern and the subsystem may be started at any point after system boot and initialization, run level 2, number 900 should be used for the start link and run level 1, number 100 for the kill link. These “Do Not Care” numbers may be used by *any* product or application without registering with Hewlett-Packard.

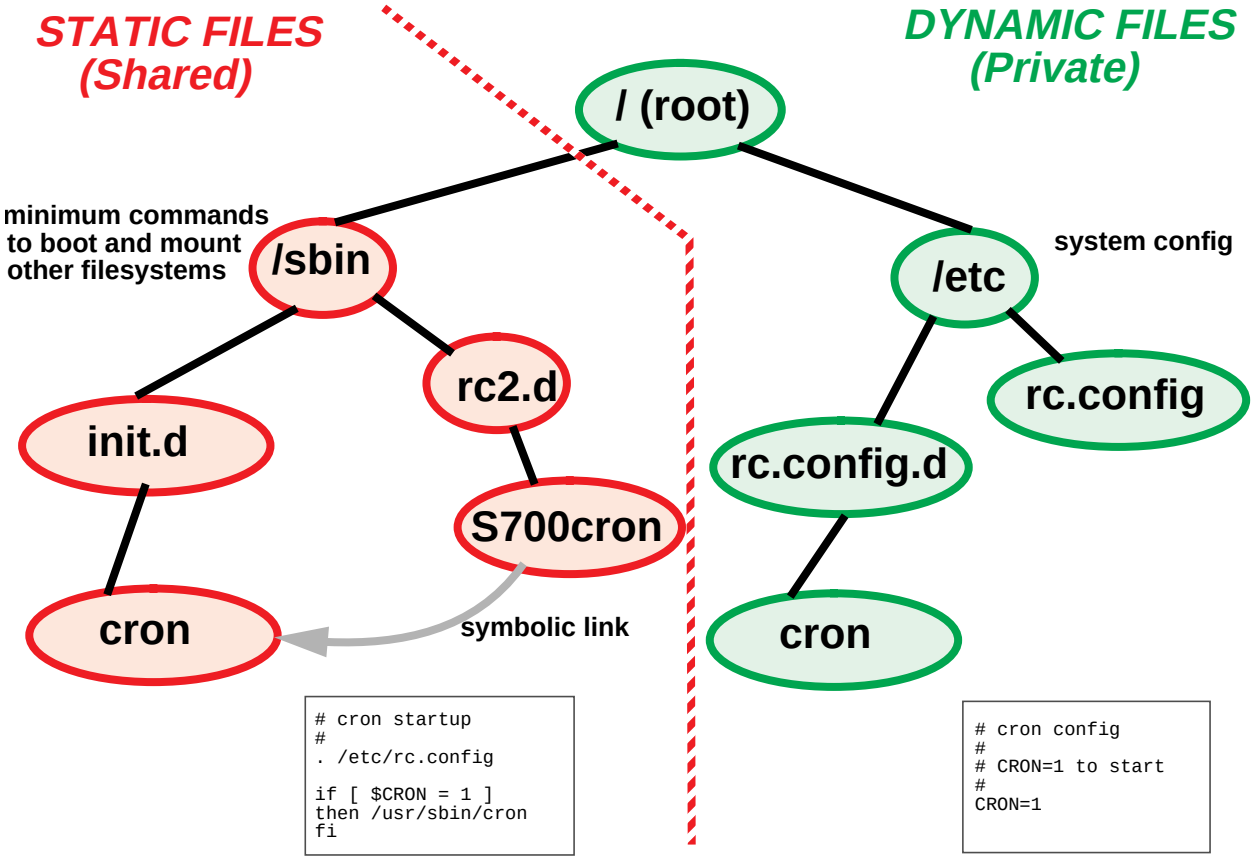
Many end-user customers may find creative ways to utilize the startup/shutdown mechanism to control their environments. In many cases, this will be realized by adding site or company specific applications layered over HP-UX. If these products are not commercially available, customer should choose their own numbers based upon the paradigm in the previous section. However, this should be done with caution when selecting specific numbers other than run level 2, #900. The numbers selected for in-house use only are not registered with Hewlett-Packard and may already be in use by a commercial product. This potential conflict may be satisfactory to organizations that do not anticipate installing many commercial applications. However, always use caution when using numbers for in-house applications, or testing.

3.2.7 An Illustrative Example

Figure 6 below depicts a simple example for the startup of cron. The relationships of the files between the static and dynamic file systems is also shown.

When entering run state 2 from a lower level, the ‘S’ scripts are executed. In the example, S700cron is a link to the cron script under /sbin/init.d. Cron will start because the configuration variable in **/etc/rc.config.d/cron** is set to 1. A value of 0 would not start cron.

FIGURE 6. Implementing CRON in the New Model



3.2.8 Guidelines for Startup/Shutdown Scripts

System execution scripts are installed in the **/sbin/init.d** directory directly from update media. Sequencer links for execution scripts are also installed into the various sequencer directories directly from update media. In the new scheme, system execution scripts are not customer editable. No provisions are made to keep newconfig versions of the scripts; any changes made by a customer to a system execution script are overwritten and lost when an update occurs.

Customers are warned of the following:

- Execution scripts must not be modified. If a script in **/sbin/init.d** is modified, the modification will be lost at the next update, when the script is overwritten with a new version.
- It is not necessary to remove scripts from the sequencer. If a script is removed from the sequencer, it will be replaced at the next update. Control variables in files within **/etc/rc.config.d** should be used to control whether a startup script is executed.
- Sequencer links must not be renamed. If a symbolic link in the sequencer directory is renamed (or renumbered), at the next update, a symbolic link with the original name will be installed. This will result in two copies of the same startup script appearing in the sequencer.

NOTE: All HP-supplied startup scripts are meant to be present in sequencer directories as installed. They should not be removed or renamed (to change the relative sequencing). Execution control should be performed via control variables in **/etc/rc.config.d/***.

3.3 Applications

The startup/shutdown specifications outlined above apply to all system components, including applications.

3.4 General Developer and User Impacts

The 10.0 Startup/Shutdown model introduces an entirely new scheme for subsystem control. Users no longer modify the large and confusing **/etc/rc*** scripts to modify the system behavior.

3.4.1 /etc/rc* Scripts

If your system or application modified or created its own **/etc/rc*** script, this will need to be changed for 10.0. Modification of the HP-supplied scripts is discouraged. Creating application (or subsystem) specific scripts is the recommended solution. Guidelines in section 3.2, "Startup/Shutdown Specifications", must be followed to ensure correct behavior.

3.4.2 Manageable Configuration and Script Files

HP is no longer supplying large **/etc/rc*** scripts. Instead, they have been separated into functional pieces, with the configuration data residing under **/etc/rc.config.d**.

Developers and users will only modify configuration files to control the system startup/shutdown behavior. Modification of the scripts under **/sbin/init.d** is not recommended.

Glossary

10FSL	Acronym for ‘10.0 File System Layout’.
Cluster	A set of one or more nodes, connected by a network, that are provided boot services by a cluster server.
Configuration data	Configuration data is information used to configure the system and includes: IP address, hostname, timezone, etc.
Execution script	An execution script is a system startup and shutdown script located in **/sbin/init.d.**
Install point	A directory into which software can be directly installed. All non-install points are potentially private directories, and must be configured on a per-host basis by execution of SD configuration scripts.
Link Files	A component of the startup/shutdown model used to link the execution scripts to a particular run-state. Link files are contained in **/sbin/rc*.d** and point to execution scripts in **/sbin/init.d.**
NFS	Network File System. An implementation of Remote Procedure Calls (RPCs) and remote mounting and access of file systems over a network.
OSF	Open Software Foundation.
PID file	A file which contains the process ID of a running application or subsystem component.
Runtime file	A working file created by an application or a subsystem. These include lock files, PID files, temporary files, state files, etc.
SD	Acronym for Software Distributor, a replacement for the HP-UX DUI update utility.
Server	The node or nodes in a cluster providing services to other nodes.
Subsystem	An operating system feature that is tightly coupled with the rest of the operating system; enough so that it is not considered an application. An example of a subsystem is NFS.
V.4	AT&T System V, release 4. Sometimes used in this paper to denote the 10.0 File System Layout.
V4FSL	Acronym for the AT&T system V Release 4 filesystem layout.