New Developments for HP-UX 10.X Filesystems

John Fenwick
Development Engineer
Hewlett-Packard Company
Enterprise Systems Division
19447 Pruneridge Avenue, MS 46T-U2
Cupertino, California USA 95014
Telephone: 011-408-447-4976
Email: fenwick@cup.hp.com

1. Introduction

With the 10.X Releases of HP-UX, filesystem support has been greatly increased. HP-UX now supports several new types of filesystems. Disk management has been made more powerful and automatic, but continues to support most previous configurations. Filesystem size (the amount of storage that can be managed within one filesystem) and file size (the size of an individual file in a filesystem) have both been increased to 128 Gigabytes. 10.X filesystem layout and bootup operations now allow for more precise control over filesystems.

Although more complex, the system administrators toolkit has been augmented with more powerful administrative commands. This paper reviews the new filesystem administration commands a system administrator will use in HP-UX 10.X.

2. Support of Multiple Filesystems

HP-UX 10.X introduced support for additional types of filesystems. Previous releases of the core HP-UX product supported the following filesystems:

hfs	High Performance (local) filesystem
nfs	Network filesystem
cdfs	CD-ROM filesystem

With HP-UX 10.X, several new types of filesystems are supported.

lofs	Loopback filesystem
vxfs	Journaled filesystem

The Loopback Filesystem allows a system administrator to logically mount one directory in the global filesystem tree onto another directory mount point. Thus, the loopback filesystem does not provide actual storage, rather it remaps the filesystem namespace. The Journaled Filesystem is a dynamic, extensible local filesystem that allocates space to files in the form of variable-sized extents which may range to several megabytes in size, and groups structural changes into transactions that are recorded in an intent log on the disk. It was first introduced in the 10.01 release of HP-UX.

The filesystem types that are defined are listed in the man page **checklist(4)** (9.X release) or **fstab(4)** (10.X release). Other types of filesystems, such as the DFS distributed filesystem, are available with the installation of additional products but are not covered in this paper.

2.1 Specification of Filesystem Type

Because multiple types of filesystem may be present on a system, the task of maintaining several types of filesystems may appear more complex. As an example, operations permitted on one type of filesystem (for example, fsck on a hfs filesystem) may be meaningless or not allowed on another type of filesystem (the fsck operation on an nfs filesystem). In HP-UX 10.X there are several ways in which the type of the underlying filesystem may be specified:

explicitly in the command line

```
% fsck -F hfs /dev/rdsk/c0t6d0
```

from a predefined entry in the file /etc/fstab:

```
% grep "/dev/dsk/c0t6d0" /etc/fstab
/dev/dsk/c0t6d0 /home/fenwick hfs defaults 0 1

% fsck /dev/rdsk/c0t6d0
```

by reading a default filesystem specification from the file **/etc/default/fs**:

```
% cat /etc/default/fs
LOCAL=hfs
```

for certain commands, by reading the contents of the filesystem superblock.

3. Filesystem Administration Utilities

In previous HP-UX releases, since fewer types of filesystem were supported, fewer commands were needed to create and maintain filesystems. With the support of multiple filesystems, these commands may contain different or conflicting command line parameters or allowed operations. Rather than complicate existing commands with the adoption of every new filesystem type, HP-UX uses a different strategy. Now, when a new filesystem type is installed, a set of filesystem administration commands is "plugged in" at the same time. These commands, such as fsck or newfs, contain the functionality to support only their specific type of filesystem, and need not have any knowledge of other filesystem types.

The administration commands for particular types of filesystems are installed in the location **/sbin/fs/<fstype>/<command>** (for commands needed at boot time) or the directory **/usr/lbin/fs/<fstype>** (for other commands). In addition, certain control and startup scripts such as **/sbin/bcheckrc** are now also specific to the underlying type of filesystem and are also located in those directories.

FIGURE 1. Filesystem Commands under /usr/lbin/fs

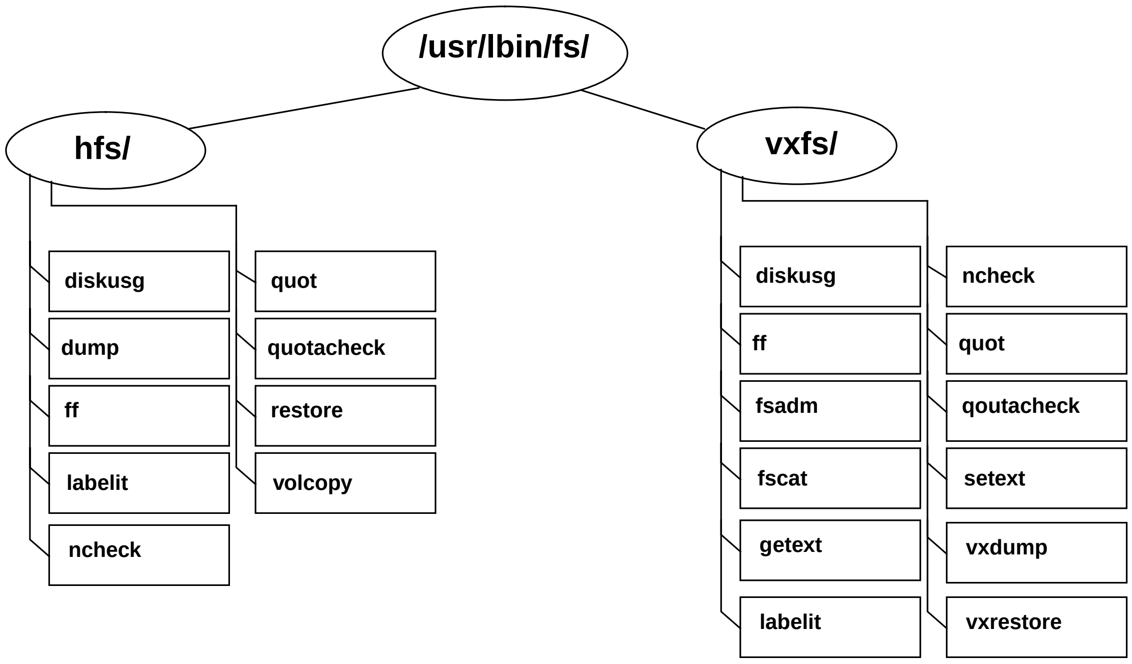
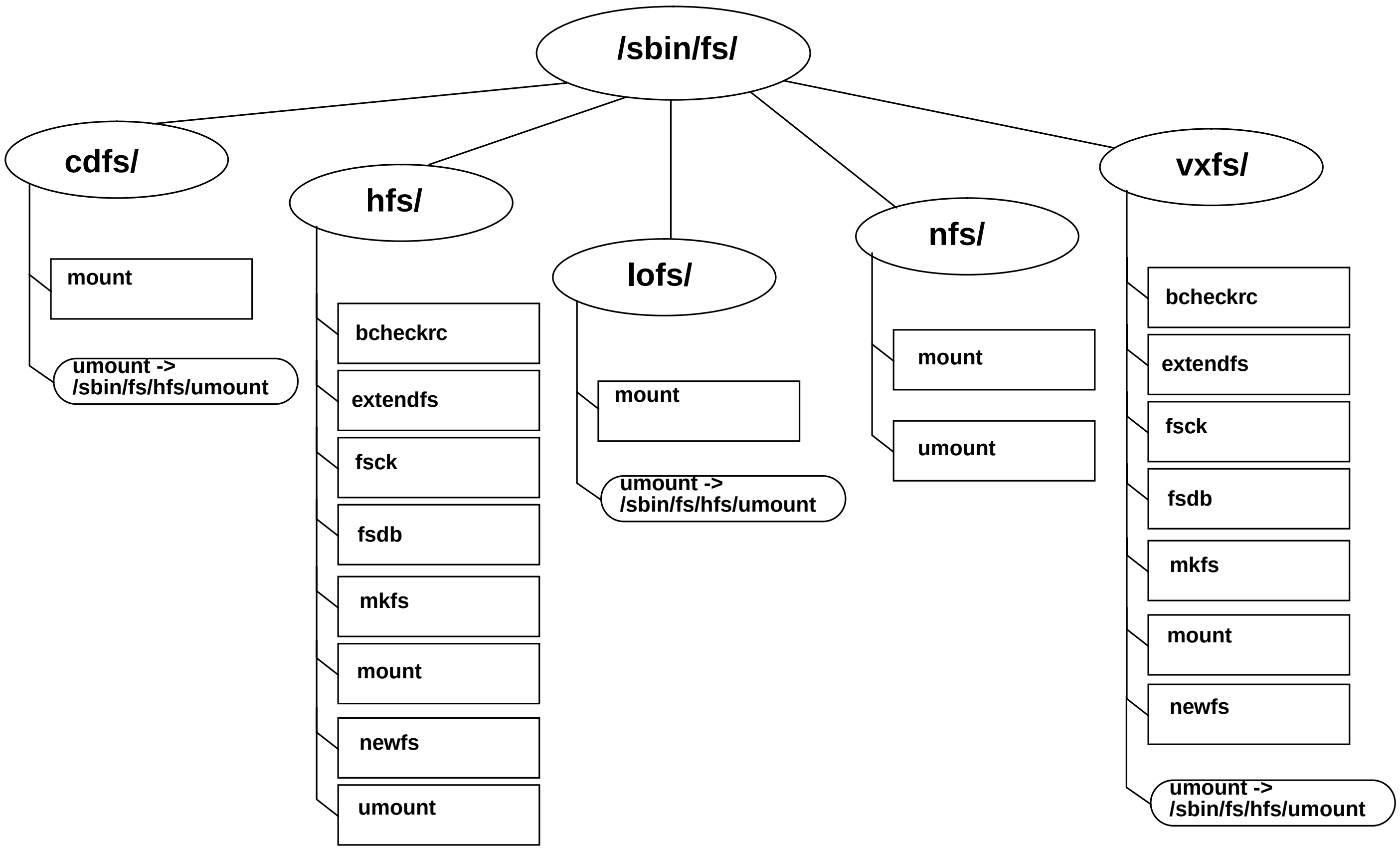


FIGURE 2. Filesystem Commands under /sbin/fs



3.1 Command paths to filesystem commands

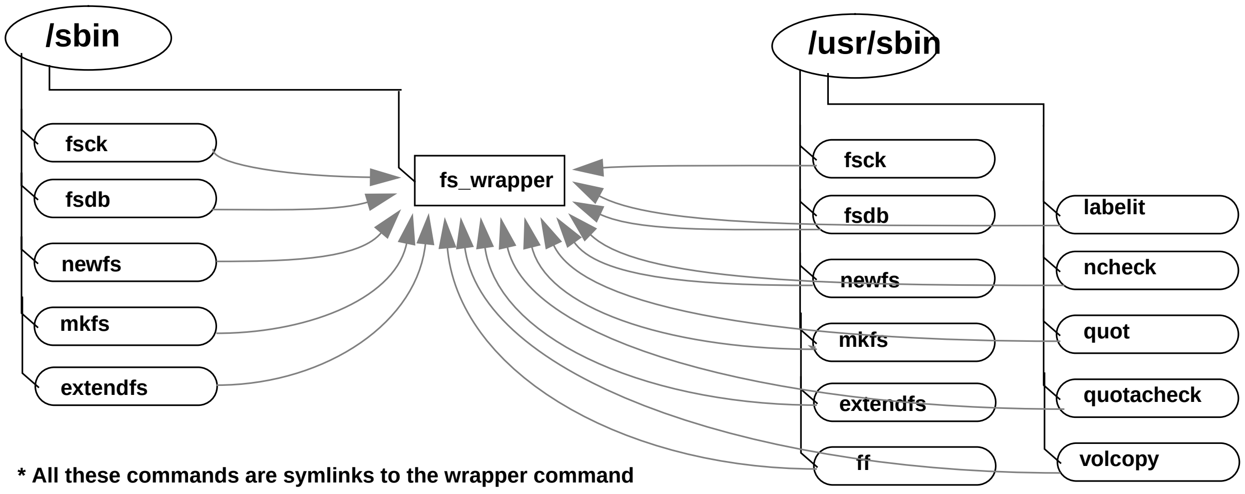
Since several different binaries for a filesystem command may now be present, there must be an easy way to reference these commands. The full path name to an instance of a command may be invoked:

```
% /sbin/fs/hfs/fsck /dev/rdsk/c0t6d0   # repair an HFS disk
```

However, many commands are now "wrapped" so that one "smart" command interface in invoked. This wrapper parses the command line and invokes the command binary appropriate for the type of filesystem. The command that would appear in a user's **$PATH** specification consists of a symbolic link to the command wrapper that invokes the appropriate binary.

The mount command is structured somewhat differently. In earlier releases of 10.X the hfs mount command would also mount filesystems of type cdfs and lofs, and would invoke a filesystem-specific version of the mount command if necessary. In later releases of 10.X the mount command is fully wrapped, and the command invoked from **/sbin/mount** will invoke a filesystem-specific version of the command. In addition, some of the system administrative scripts such as **/sbin/bcheckrc** have also become filesystem-specific. If a new filesystem type is installed on a system, new versions of commands specific for this filesystem type will be automatically installed and invoked when appropriate.

FIGURE 3. Figure 3: Filesystem Commands and the "smart" command interface.



3.2 Manual pages for multiple filesystems

Manual page definitions for commands that are filesystem-specific may also be delivered. The man pages for such commands have the following name conventions:

```
% man 1m <command>         # generic or wrapped command
% man 1m <command>_<fstype># filesystem-specific version
```

For example, the man page definitions for the fsck command are found as follows:

```
% man 1m fsck
   returns manual page for fsck wrapper
```

```
% man 1m fsck_hfs
   returns manual page for fsck operations on hfs disks
% man 1m fsck_vxfs
   returns manual page for fsck operations on vxfs disks
% man 1m fsck_[nfs,cdfs,lofs]    # not defined or meaningful
No manual entry for fsck_[nfs,cdfs,lofs]
```

For a filesystem operation that is not meaningful, there is no associated manual page definition.

4. Disk Management

4.1 File system creation

4.1.1 newfs(1m)

With the 10.X release, the newfs(1m) command was extensively modified. Some of the new features of this command are as follows:

- newfs automatically figures disk size and disk geometry - it does not require a disk_type to be specified on the command line or have a corresponding entry in the file **/etc/disktab**
- newfs does not automatically reserve space for a swap or boot partition, and this space must be explicitly reserved if desired

Note: some of these capabilities were present in earlier releases of the System Administrator (SAM).

As an example, consider a raw device file corresponding to a 660 MB disk. One may create a new file-system, reserving space for 40 Megabytes for swap, with the command:

```
% newfs /dev/rdsk/c0t6d0 -R 40
```

Note that the filesystem type is not specified with “-F <fstype>”. In this case, the command will use the default file system type defined in the file /**etc/default/fs**.

Note also that the default operation of newfs_hfs is to not reserve any space for either a swap partition or for a boot partition on the disk. If one is creating a filesystem that is intended to be used as a boot disk and to have a swap partition one must explicitly reserve space for boot and swap as follows:

```
% newfs /dev/rdsk/c0t6d0 -B -R 40
```

The disk descriptions for whole-disk filesystems (S700 type) are still delivered in **/etc/disktab**, but these descriptions need not ordinarily be required.

4.1.2 mkfs(1m)

The command mkfs(1m) has been modified in a manner similar to newfs. mkfs does not need to use the disk_type description from the file **/etc/disktab**. The optional parameters to mkfs (for numbers of sectors and tracks, block and fragment size, cylinder information, etc.) can still be entered on the mkfs command line or in a prototype file specification if necessary.

Since mkfs with no optional parameters will configure the filesystem in a way similar to newfs, one may simply use the mkfs command to build a filesystem similar to the one above as follows:

```
        % mkfs /dev/rdsk/c0t6d0 (size_in_blocks)
```

where size_in_blocks is: (disk_size - 40) / **DEV_BSIZE**

4.2 Disk partitioning

4.2.1 Series 800 Hard Partitions

In previous releases of HP-UX for the Series 800, disks could be configured with a set of hard partitions; these partitions could then be mapped to different directory mounts in the filesystem tree. The **/etc/checklist** file from a S800 9.X system with one disk divided into a number of partitions is listed below.

```
        /dev/dsk/c0d0s10 /          hfs         rw 0 1
        /dev/dsk/c0d0s1 swap        ignore      sw 0 0
        /dev/dsk/c0d0s4 /tmp        hfs         rw 0 2
        /dev/dsk/c0d0s5 /usr        hfs         rw 0 2
        /dev/dsk/c0d0s3 /users      hfs         rw 0 2
```

The default disk partitioning in 10.X, for both the Series 800 and Series 700, is for a single “whole-disk” partition. If a configuration other than this is required, the Logical Volume Manager on both the S700 and S800 allows for flexible partitioning and management of disk space on a single drive or on multiple drives.

Hard partitions on single disks are still supported in 10.X. A field (bits 28-31) in the disk device minor number is used to signify the section number. For the above example, the 9.X disk hard partitions could be mounted to a 10.X system using these device special files with associated minor numbers.

```
        /dev/dsk/c0t0d0s10   0x00000a
        /dev/dsk/c0t0d0s4    0x000004
        /dev/dsk/c0t0d0s5    0x000005
        /dev/dsk/c0t0d0s3    0x000003
```

By mounting a hard-partitioned disk configured in a previous version of HP-UX through these drivers, the old disk partitions and filesystems are preserved while mounted to a 10.X system. This may also prove useful to the System Administrator to allow one to mount a hard-partitioned disk to a 10.X system for filesystem maintenance and repair. For example, suppose one were maintaining an S800 9.X system with hard partitions, and the root partition became corrupted. By mounting the 9.X disk to a 10.X system through this interface, one could mount the corrupted 9.X root disk onto a 10.X system and perform filesystem repair (or recovery from a backup media) from the 10.X host. Alternatively, the use of these “compatibility-mode” drivers allows a system administrator to mount filesystems originally created on a S800 system onto a S700 computer, even if the filesystems are of the old “S800 hard partition” type.

4.2.2 Series 700 Software Disk Striping

In the 9.X release of HP-UX for S700 systems, Software Disk Striping (SDS) allowed a system administrator to logically partition a single disk into a number of different partitions in which each could contain a separate filesystem, or to create a number of filesystems each of which could span several physical disks. The SDS product has been obsoleted in HP-UX 10.X; the Logical Volume Manager (LVM) is now supported on both the S700 and S800 and provides superior functionality. For existing SDS installations, the utility **sdstolvm** converts an SDS installation into an LVM installation.

5. Large Filesystems and Large Files

5.1 Large Filesystem Support in HP-UX

Starting with HP-UX Release 10.10, the maximum size of filesystems has been increased from 4 Gigabytes to 128 Gigabytes. This size applies to filesystems of type hfs and vxfs.

This change is of interest to System Administrators, and is mostly transparent to the average user. The changes to HP-UX to support this larger filesystem size consist of internal changes to the HP-UX kernel, and changes to certain type of filesystem administration commands. These commands include the following:

• commands to create filesystems	newfs, mkfs
• commands to modify filesystems	fsck, fsdb, etc
• commands that report filesystem statistics	df, bdf, dumpfs, etc.

5.2 Large File support in HP-UX

Starting with HP-UX Release 10.20, the maximum size of individual files within a filesystem has been increased from 2 Gigabytes to 128 Gigabytes. This larger file size is supported on HFS and VxFS (V3) filesystems.

Changes to support large files are more visible to the user and application developer than are changes to support large filesystems. A new set of filesystem manipulation library calls and a new compile option are used to determine if a filesystem call uses a 32-bit or 64-bit interface. Code that has been developed that expects file pointers to be of a certain size may need to be modified to support the new larger pointers. To protect installed applications that have not been modified to support larger file pointers, individual filesystems may be configured as supporting “largefiles” or “nolargefiles”. This filesystem “feature” may be set with the fsadm(1m) command.

In HP-UX only certain types of commands have been modified to directly support large files. Examples of the commands that have been modified to support large files include the following:

- accounting and quota commands
- hfs and vxfs filesystem administration commands
- fbackup/frecover
- other selected commands, such as:
  cat, cp, mv, ls, rm, rmdir, chmod, chown, chgrp, du, df, dd, od
  grep, awk, sort, uniq, strings, cmp, csplit, paste, cut, wc, shells

6. Root and Boot Filesystems

In earlier releases of HP-UX the root filesystem had to be of type hfs (or nfs for the special case of 10.X NFS Diskless Clustering). For later releases of HP-UX 10.X, it will be possible to support different types of filesystems as the root filesystem for the system.

In these configurations, a distinction is made between the Boot Filesystem (read to bring in the kernel and other files at boot), and the Root Filesystem (the global root of the filesystem directory tree).

6.1 Boot Filesystem

The Boot Filesystem contains the system kernel and configuration files required in the boot process. These files, along with a few other files required for software installation and kernel configuration, are found in the directory /stand in the 10.X Filesystem Layout:

```
% ls -1F /stand
bootconf
build/
ioconfig
kernrel
system
vmunix*
```

A separate local filesystem for the **/stand** directory must be of type hfs to be readable by the Initial System Loader (ISL). ISL will look for the kernel along the paths "**/stand**" (for single filesystem disks) and "**/**" (for a system with **/stand** mounted to a separate filesystem) on an hfs disk at boot time.

NFS Diskless Clustering also supports the concept of separate root and boot filesystems. In an NFSD system, certain files from **/stand** are initially read by the client via TFTP transfer, and **/stand** is NFS mounted later in the boot process. The root filesystem ("**/**') is also NFS mounted to the client.

6.2 Root Filesystem

The Root filesystem is the global root of the HP-UX filesystem space. The kernel mounts the root filesystem to "**/**" as part of the boot process.

6.3 Mount of boot filesystem /stand directory

If the boot directory **/stand** is mounted as a separate filesystem, the directory "/stand" must exist in the root directory as the mount point. The filesystem to be mounted is checked for consistency and is mounted into the filesystem tree as part of user space initialization. Just as with mounts of other filesystems, **/stand** must be listed as an entry in the file **/etc/fstab**. An example entry would be as follows:

```
# fstab entries - mountable filesystems
/dev/vg00/lvol1 /stand hfs defaults 0 1
```

7. Root Filesystem Contents - System Administration at Boot Time

At boot time, we may need to mount several filesystems of different types. A system failing at some point in startup may be left in a partially working state. In such a state only certain commands and subsystems may be available. Additional restrictions on the operations possible at boot time exist in installations where the **/usr** directory is mounted to a separate filesystem.

7.1 Filesystem Checking and Mounting at Boot Time

HP-UX uses a type of bootstrap approach to the mounting of filesystems at startup time. Each filesystem is checked for consistency (typically with the fsck operation) before it is mounted. This helps ensure that all files read into the system are valid and helps prevent system errors caused by faulty or missing files. The system administrator is given an opportunity to interact with HP-UX before any changes or repairs are made to the filesystems. The root filesystem is the first filesystem mounted and is key to successful operation; other filesystems are mounted into the global filesystem tree after root has been mounted. The following steps are performed in the HP-UX kernel and command scripts to check and mount filesystems.

kernel (init_main)	mount root disk read-only to read key files such as sh, fsck, pre_init_rc
kernel (init_main)	execute pre_init_rc, which does fsck -P of root disk
kernel (init_main)	unmounts root disk, and then mounts it again read-write
ioinitrc (inittab line 1)	fsck -P then mount if clean /stand filesystem
bcheckrc (inittab line 2)	run bcheckrc_<fs> for multiple filesystem types
	test local filesystems; if not clean fsck -P local filesystems
	if root modified then reboot -n without disk update
	if still not clean: run a shell for manual fsck
localmount (runlevel 1)	mount local filesystems: mountall -l -m
	enable filesystem quotas if configured

7.2 HP-UX run levels at boot time

For the System Administrator trying to perform maintenance at system boot, the run levels of interest are levels S, 1 and 2.

7.2.1 Run level S

When entering state S on bootup (the user entered a flag to the boot loader), the init process will attempt to read **/etc/inittab** but will process entries only of type “sysinit”. In particular, the init process does not invoke the 10.X run level sequencer **/sbin/rc**. A shell is run at the system console and the system is in single user state. This state may be used for system administration, since the system is quiescent with a minimum of services and filesystems active. In run level S, one can create and repair filesystems as necessary to prepare the system for full functionality in the multi-user run levels. In a system where the **/usr** directory is a separately mounted filesystem, files under **/usr** are not available at this time.

If entering state S from multi-user state (the command “init S” is entered), the same steps are taken: i.e., a shell is started at the console. No processes are killed, no attempt is made to put the system into a quiescent state. So entering runlevel “S” from multi-user mode is not a useful operation in HP-UX.

7.2.2 Run level 1

In HP-UX 10.X, the primary use for run level 1 is to mount and repair local filesystems (such as the **/usr** directory). If the /usr directory has been mounted to a separate filesystem in run level 1, many more commands and subsystems become available in this run level than are available in run level S. Some additional work is done entering run level 1, such as setting key system parameters, saving a system core image if present on a dump device, and starting daemons such as the swapper and syncer.

Since run level 1 is not a multi-user state, it is a useful state in which to do system administration. However, run level 1 is different from run level S: more entries of **/etc/inittab** may be invoked, the master sequencer **/sbin/rc** is run to transition the system from run level S into run level 1, and local filesystems are mounted.

7.3 HP-UX Startup Strategy

As with the mounting of filesystems, HP-UX uses a type of bootstrap approach to other parts of the startup process. At earliest stages of user-space startup, a minimum of services may be available, and a minimum of commands may be available on the root volume. In particular, the **/usr** filesystem may not be mounted. Under /usr lies the bulk of the operating system commands, the networking commands, NLS message catalogs, the dynamic loader (**/usr/lib/dld.sl)** and libraries including shared libraries (**/usr/lib/*.sl**).

This approach results in some robustness for the root filesystem. A minimum of functionality is required to perform some basic filesystem administration commands. There is no single point of failure due to a failed or missing dynamic loader or library file. Files on root disk are the minimum needed for system boot, mounting of other filesystems, and minimal system repair.

7.4 Constraints on the root filesystem

The root filesystem is mounted by the kernel; additional filesystems are typically mounted at the early stages of user space startup. The following directories must be located on the root filesystem:

```
/        (root)
/dev     device special files
/etc     configuration files
/sbin    commands to boot, mount filesystems, repair system
```

The filesystem subtrees beneath these directories are optionally located on other filesystem mounts and if so are typically mounted in run level 1:

```
/usr     the bulk of operating system commands and utilities
/var     dynamic log, spool, and configuration files
/opt/*   installed applications
```

7.5 Commands available on the Root Disk

Currently all the binaries HP-UX delivers on the root volume are built completely statically linked. In a small number of cases, a binary in **/sbin** will also be present in **/usr/[s]bin**; this alternate version if present is typically built dynamically linked.Commands on the root volume must be statically linked, because neither the dynamic loader nor the shared libraries may be available. The contents of the **/sbin** directory are restricted. If the functionality of the command on the root volume is limited, the command is duplicated and there exist separate **/sbin/** and **/usr/[s]bin/** copies:

/sbin copy	statically linked
/usr/[s]bin copy	typically dynamically linked

Root volume commands are of the following types. A complete listing is given in Table 1:

startup and shutdown:

kernel, init, rc scripts, rc.config files, reboot, shutdown

I/O system initialization and hardware initialization:

ioinit, insf, ioscan, download, dasetup, eisa_config, mtinit shell (POSIX-sh)

filesystem mount and repair

mount, umount, mknod, fsck, fsclean, fsdb, mkfs, newfs, mkboot

A small set of utilities:

ls, cat, link, unlink, mv, ln, rm, stty, awk, savecore, dmesg, mkdir, chmod, chown

Recovery utilities:

tar, pax, frecover

Table 1: Root filesystem commands and duplicates

/sbin entry	/usr/sbin	/usr/bin
SnmpAgtStart.d/		
awk		/usr/bin/awk
bcheckrc		
cat		/usr/bin/cat
chmod		/usr/bin/chmod
chown		/usr/bin/chown
dasetup		
date		/usr/bin/date
dmesg	/usr/sbin/dmesg -> /sbin/dmesg	
download		
eisa_config		
extendfs	/usr/sbin/extendfs -> /sbin/fs_wrapper	
false		/usr/bin/false
frecover	/usr/sbin/frecover	
fs/		
fs_wrapper		
fsck -> /sbin/fs_wrapper	/usr/sbin/fsck -> /sbin/fs_wrapper	
fsclean	/usr/sbin/fsclean -> /sbin/fsclean	
fsdb -> /sbin/fs_wrapper	/usr/sbin/fsdb -> /sbin/fs_wrapper	
fstyp	/usr/sbin/fstyp -> /sbin/fstyp	
init		
init.d/		
insf (lssf,mksf,rmsf)	/usr/sbin/insf -> /sbin/mksf	
ioinit	/usr/sbin/ioinit -> /sbin/ioinit	
ioinitrc		
ioscan	/usr/sbin/ioscan -> /sbin/ioscan	
is_local_root		
lib/		
line		/usr/bin/line
link (unlink)	/usr/sbin/link -> /sbin/link	
ln		/usr/bin/ln
ls		/usr/bin/ls
lvchange (23 links)	/usr/sbin/lvchange (23 links)	

Table 1: Root filesystem commands and duplicates

/sbin entry	/usr/sbin	/usr/bin
lvmrc		
mkboot		
mkdir		/usr/bin/mkdir
mkfs -> /sbin/fs_wrapper	/usr/sbin/mkfs -> /sbin/fs_wrapper	
mknod	/usr/sbin/mknod -> /sbin/mknod	
mount	/usr/sbin/mount -> /sbin/mount	
mountall	/usr/sbin/mountall -> /sbin/mountall	
mtinit		
mv		/usr/bin/mv
newfs -> /sbin/fs_wrapper	/usr/sbin/newfs -> /sbin/fs_wrapper	
passwd		/usr/bin/passwd
pax		/usr/bin/pax
powerfail		
pre_init_rc		
rc		
rc.utils		
rc0.d/		
rc1.d/		
rc2.d/		
rc3.d/		
rc4.d/		
reboot	/usr/sbin/reboot -> /sbin/reboot	
restore	/usr/sbin/restore -> /usr/lbin/fs/hfs/restore	
rm		/usr/bin/rm
savecore		
set_parms		
set_parms.d/		
set_parms.util		
sh		/usr/bin/sh
shutdown	/usr/sbin/shutdown -> /sbin/shutdown	
stty		/usr/bin/stty
tar		/usr/bin/tar
true		/usr/bin/true
ttytype		/usr/bin/ttytype
umount	/usr/sbin/umount -> /sbin/umount	
umountall	/usr/sbin/umountall -> /sbin/umountall	

7.6 Limitations on library entry points from root volume commands

Future versions of HP-UX may have restrictions on the library entry points that are available to commands on the root volume. In particular, there may be restrictions on library entry points that may require underlying functionality that is not available at all times. These restrictions should only apply to partners developing their own binaries that must be located on the root volume. Please contact HP if your application may be required to operate under these conditions.