Insider Tips: The Unix filesystem tree

GETTING ORGANIZED

Unix systems organize files in a hierarchical filesystem tree. A system of naming conventions defined in the Filesystem Hierarchy Standard (FHS) helps admins find their way around.

BY MARC ANDRÉ SELIG

Even a minimal Linux installation writes thousands of files and directories to disk. The rise of different Unix dialects led to a number of completely different designs for organizing files in a meaningful way on disk. For example, some admins named their users' home directories */usr/home/Name*, whereas others preferred */Users/Name*. A mailbox might be */usr/mail/Name* on one machine but */var/spool/mail/Name* on another. In this case, diversity was a drawback.

In contrast to Windows, for example, Unix's modular design requires administrators to select a single program for each task from a choice of many possible options and to replace this program with a backwards-compatible upgrade if a new version becomes available. For example, if an admin needs to replace the Mail Transfer Agent (MTA), the new version should be capable of locating and handling existing messages.

Unless you want to define the paths yourself, conforming to standards is a preferred approach. Standardized directory structures also simplify exchanging files between different machines. Some administrators make life easier for themselves by performing major software installations just once. All the machines on the LAN then use a network filesystem such as NFS to mount the program directories. Of course, this only works if critical programs and files are in the same place on each machine.

The Linux community started to work on formally standardizing the filesystem as early as the fall of 1993 (before the kernel 1.0 release in March 1994). The idea behind standardizing the filesystem was to be able to port software from any Unix variant to any Linux distribution with the least possible effort. The FSSTND (Filesystem Standard) project followed the Filesystem Hierarchy Standard FHS [1], which is still the standard for all distributions today. Although the FHS specifications provoked arguments in many cases, the mere fact that they existed brought about the standardization that many Linux distributions have come to rely on.

Basics

Each Linux/Unix computer needs a number of core files and directories. Even if the computer retrieves everything else from the network, this core group of files must be available from the outset. In the case of systems that comprise multiple, physical partitions, these files reside on the boot partition. (When a computer boots, it can retrieve these files from a server, but this needs to happen before *init* starts initializing the system.)

The root filesystem contains a fully working Unix system, and this offers a number of benefits: if the network interface card driver fails after a kernel update, it might be impossible to activate some parts of the filesystem. The admin needs access to the root filesystem where the troubleshooting tools reside. The same thing applies to restoring damaged files from a backup medium: everything an admin needs for this chore should reside in the root area.

The toolkit should include the user's favorite shell and required libraries, a simple editor, and software for accessing

other parts of the system such as *fdisk*, the *mkfs* family, and *mount*. The Filesystem Hierarchy Standard specification [1] has a complete list.

Besides the root directory */*, the root filesystem area includes the */etc* directory with its configuration files, */bin* and */sbin* with the most important programs, */lib* for dynamic libraries and kernel modules, and */dev* for device files. The root user's home directory, */root*, is also often found on this partition.

/boot plays a special role; on a modern Linux system, the kernel, a ramdisk image with critical kernel modules, and the configuration files for the bootloader (Grub or Lilo) are stored here. Although these files are part of the core Linux system, they do not need to reside on the partition with the root filesystem. After all, the bootloader does not bother about filesystems while loading the kernel.

Resources

Endusers vary rarely run programs on the root partition, as most application software is located in */usr*. People who are familiar with Unix will read *usr* as meaning "Unix System Resources," although it is commonly referred to as the "user" directory. */usr* also has many of the subdirectories available on the root filesystem. There is a */usr/bin* to match */bin*; system programs reside both in */sbin* and */usr/sbin*, and there is */usr/lib* for dynamic libraries. According to the FHS rules, there should not be a */usr/etc*. The configuration data for programs in */usr* should reside in */etc*.

/usr also has a number of additional subdirectories: source code is stored in */usr/src*, */usr/include* stores include files for various programming languages, and */usr/share/doc*, */usr/share/man* and */usr/share/info* store documentation files for installed software packages.

To be FHS compliant, additional software can be stored both in */usr* and in */opt*. Whereas */usr* organizes files by function (binaries are stored in */usr/bin*, libraries in */usr/lib*), */opt* organizes files by vendor or software package. Solaris is an extreme example of this cross-platform convention with its collection of various */opt/SUNW** packages. Linux

Listing 1: Swapping out /usr/share

```
01 #!/bin/sh
02 mke3fs /dev/sdc1
03 mount /dev/sdc1 /mnt
04 cd /usr/share
05 tar cf - . | (cd /mnt && tar xpf -)
06 umount /mnt
07 mount -o remount,rw /usr      # falls /usr
   read-only ist
08 rm -rf /usr/share/*
09 mount -o remount,rw /         # falls /
   read-only ist
10 echo '/dev/sdc1 /usr/share ext3  defaults 1 3'
   >>/etc/fstab
11 mount /usr/share
12 exit
```

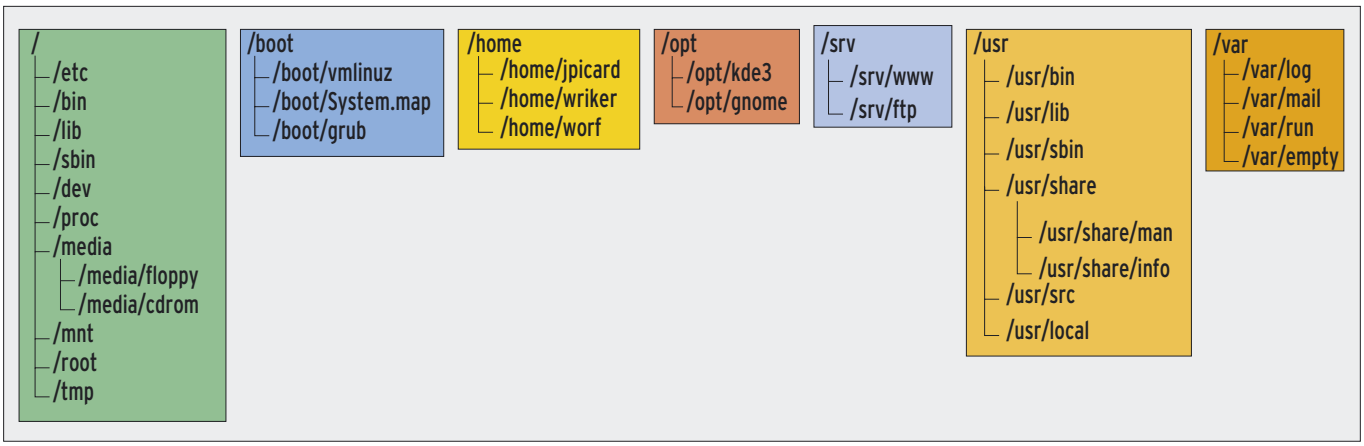

Figure 1: The filesystem tree on a typical Linux machine. For good performance and flexibility, admins should distribute data across multiple hard disk partitions.

often has installations with */opt/kde3* or */opt/gnome*. But to use packages that you install in */opt* in the shell, you will need to set your *PATH* variable.

The files in */usr* and */opt* are fairly static by nature. It makes sense to mount these two directories as read only after initializing the system; this avoids accidental or deliberate modifications. And although it is important to have a complete backup, you can normally do without a backup of static partitions [2].

/usr tends to fill up with data, especially on older private computers. In this case, admins need to consider moving subdirectories from */usr* out to partitions of their own, as shown in Listing 1. The script creates a file system on an empty partition, */dev/sdc1*, and then moves the content from */usr/share* to this partition.

Home, Sweet Home

The home directories on BSD-based systems traditionally reside below */usr/home*. Linux uses */home* instead. On local networks, admins very often use an automounter to map */home* to a network directory, which is mounted dynamically from a fileserver when a user logs on.

The data directories of services such as HTTP or FTP servers are also quite commonly assigned their own directories. Distributions have completely different approaches to mountpoints, although the FHS stipulates that this data should reside below a */srv* directory. Suse actually does this, allowing the Apache document root to point to */srv/www*.

Many administrators find this hard to get used to. For example, Debian is extremely proud of being FHS compliant, although it uses */var/www* as the document root; Red Hat Enterprise Linux adopts a similar approach. The data directory for the FTP daemon typically resides below */home/ftp*.

Software Builds

If you build and install your own software on a Linux system, you should not use one of the directories we have mentioned previously. */usr/local* gives you a separate structure for this purpose. Scripts or programs you have written yourself should reside below */usr/local/bin*, whereas the documentation should be in */usr/local/man*. You can decide for yourself whether you want to put the source code in */usr/local/src* for all to read or keep it in your own home directory.

It also makes sense to put */usr/local* and */home* on separate partitions. If you move to another distribution, a simple mount command or a new entry in */etc/fstab* will give you your files back in next to no time. If you prefer to avoid creating too many partitions, you could just create a partition for */home* and use a symlink to */home/local* in */usr/local*.

Static and Variable Data

A running Unix system produces a large number of rapidly changing files that are collected in the */var* (variable data) directory. There are two major benefits to this approach: first, it separates the volatile data from the read only data on the */usr* partition. Additionally, logfiles can grow to a considerable size, depending on the amount of use a system gets. In case of unexpectedly heavy load, Apache or firewall logfiles can really explode. If the logfile fills up the root partition, the system will start to do strange things because any number of programs are unable to write important temporary files. In other words, it is a good idea to put */var* on a separate partition.

One subdirectory of */var* that is extremely important is */var/log* (this is the place where logfiles are normally stored); */var/mail* holds the user mailboxes, although older Linux systems use */var/spool/mail*. */var/run* has the PID files with the process IDs of active daemons that some inter process communication (IPC) programs use [3]. */var/empty* is an empty home directory for the privileged part of the SSH daemon, for example.

The Rest

The root filesystem also has mountpoints for other filesystems, including */proc*, a virtual filesystem that provides a number of kernel variables. Removable media use subdirectories below */media*, such as */media/floppy* or */media/cdrom*. According to the standard, */mnt* should only be used to mount filesystems for temporary use, for example, for troubleshooting a machine or for short-term access to a network filesystem. ■

INFO

[1] Filesystem Hierarchy Standard: *http://www.pathname.com/fhs/*

[2] Marc André Selig, "Backups": Linux Magazine 05/05, p. 66.

[3] Marc André Selig, "Hand Signals: Signal-based interprocess communication": Linux Magazine 05/04, p. 61.