SECTION 5 Manage Directories and Files in Linux

In this section you learn about the structure of the Linux file system and the most important file operation commands for working at the command line.

Objectives

1. Understand the Filesystem Hierarchy Standard (FHS)
2. Identify File Types in the Linux System
3. Change Directories and List Directory Contents
4. Create and View Files
5. Manage Files and Directories
6. Find Files
7. Search File Content
8. Archive, Back Up, Compress, and Decompress Files

Objective 1 Understand the Filesystem Hierarchy Standard (FHS)

The file system concept of Linux (and, in general, of all UNIX systems) is considerably different than that of other operating systems.

To understand the concept of the Linux file system, you need to know the following:

- The Hierarchical Structure of the File System
- FHS (Filesystem Hierarchy Standard)
- Root Directory /
- Essential Binaries for Use by All Users (/bin/)
- Boot Directory (/boot/)
- Device Files (/dev/)
- Configuration Files (/etc/)
- User Directories (/home/)
- Libraries (/lib/)
- Mount Points for Removable Media (/media/*)
- Application Directory (/opt/)
- Home Directory of the Administrator (/root/)
- System Binaries (/sbin/)
- Data Directories for Services (/srv/)
- Temporary Area (/tmp/)
- The Hierarchy below /usr/
- Variable Files (/var/)
- Process Files (/proc/)
- System Information Directory (/sys/)

- Mount Point for Temporarily Mounted File Systems (/mnt/)
- Directories for Mounting Other File Systems

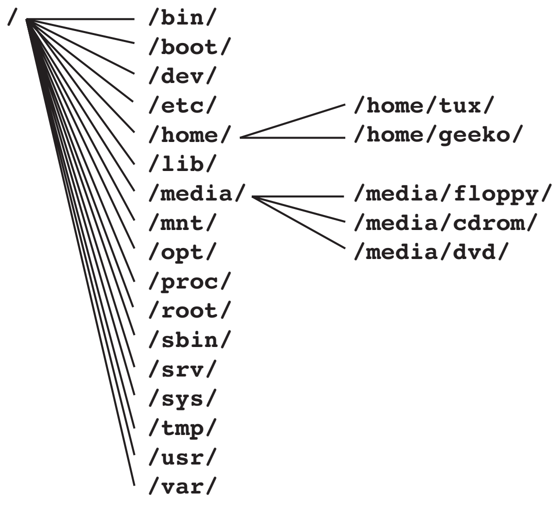
The Hierarchical Structure of the File System

The file system concept of Linux involves a hierarchical file system that can be depicted in the form of a tree.

This tree is not limited to a local partition. It can stretch over several partitions, which can be located on different computers in a network. It begins at the root, from where the name for the system administrator comes, and branches out like the branches of a tree.

The following shows part of a typical file system tree:

Figure 5-1



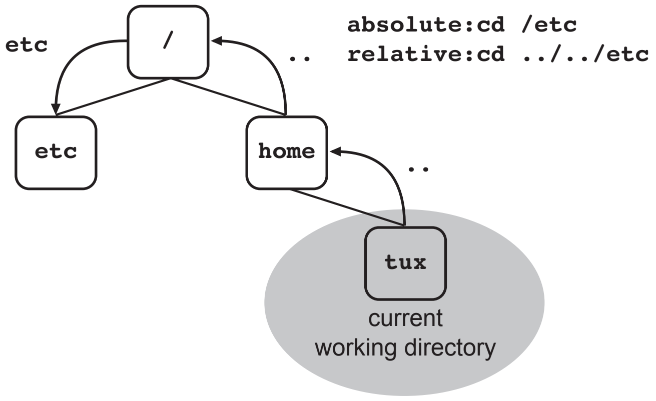
A file in this directory tree is uniquely defined by its path. A path refers to the directory names that lead to this file

The separation character between individual directory names is the slash ("/"). The path can be specified in two ways:

- As a *relative path* starting from the current directory
- As an *absolute path* starting from the root of the entire file system tree

The absolute path always begins with a slash (/), the symbol for the root directory, as in the following:

Figure 5-2



Sometimes it is necessary to specify the absolute path because certain files can only be uniquely addressed in this way. The length of the path cannot exceed 4096 characters, including the slashes.

FHS (Filesystem Hierarchy Standard)

The structure of the file system is described in the Filesystem Hierarchy Standard (FHS). FHS specifies which directories must be located on the first level after the root directory and what they contain.

FHS does not specify all details. In some areas, it allows leeway for your own definitions. FHS defines a two-layered hierarchy:

- The directories in the top layer (immediately below the root directory "/")

- As a second layer, the directories under /usr/ and /var/

You can find information about FHS at http://www.pathname.com/fhs/ on the Internet.

Root Directory /

The root directory refers to the highest layer of the file system tree. Only directories are located here, not files. When the system is booted, the partition on which this directory is located is the first one mounted.

Because the kernel cannot complete all the tasks of the operating system, all programs that are run on the system start must be available on this partition; they cannot be located on another partition.

The following directories always have to be on the same partition as the root directory: /bin/, /dev/, /etc/, /lib/, and /sbin/.

Essential Binaries for Use by All Users (/bin/)

The directory /bin/ contains important executable programs that are required when no other file systems are mounted, such as all programs necessary for the system start.

These include the various shells, the most important commands for working with files, and several commands for system analysis and configuration.

The following table provides an overview of the contents of the /bin/ directory:

Table 5-1

File	Description
/bin/bash	The bash shell
/bin/cat	Display files
/bin/cp	Copy files
/bin/dd	Copy files byte-wise
/bin/gzip	Compress files
/bin/mount	Mount file systems
/bin/rm	Delete files
/bin/vi	vi editor

Boot Directory (/boot/)

The directory /boot/ contains static files of the boot loader (GRUB or LILO). These are files required for the boot process (with the exception of configuration files).

The backed-up information for the Master Boot Record (MBR) and the system map files are also stored here. These contain information about where exactly the kernel is located on the partition. This directory also contains the kernel.

According to FHS, however, the kernel can also be located directly in the root directory.

Device Files (/dev/)

Each hardware component existing in the system (such as hard drive partitions, CD drives, printer, and mouse) is represented as a file in the directory /dev/. (An exception are network cards, which are not represented by a device file.)

The hardware components are addressed via these files by writing to or reading from one of these files. Two kinds of device files are included:

- Character-oriented device files (for devices working sequentially, such as printer, mouse, or tape drive)
- Block-oriented device files (such as floppy disks and hard drives)

The connection to device drivers in the kernel is implemented via numbered channels, which correspond to the number of the device driver in question. These are referred to as *major device numbers*.

A driver might be responsible for several devices of the same type. To distinguish between these devices, the *minor device number* is used.

Instead of the size of the files, these two numbers are displayed (the files do not occupy any space on the hard drive):

```
da10:~ # ls -l /dev/hda*
brw-rw---- 1 root disk 3, 0 Mar 22 06:12 /dev/hda
brw-rw---- 1 root disk 3, 1 Mar 22 06:12 /dev/hda1
brw-rw---- 1 root disk 3, 10 Mar 22 06:12 /dev/hda10
brw-rw---- 1 root disk 3, 11 Mar 22 06:12 /dev/hda11
brw-rw---- 1 root disk 3, 12 Mar 22 06:12 /dev/hda12
brw-rw---- 1 root disk 3, 13 Mar 22 06:12 /dev/hda13
brw-rw---- 1 root disk 3, 14 Mar 22 06:12 /dev/hda14
brw-rw---- 1 root disk 3, 15 Mar 22 06:12 /dev/hda15
brw-rw---- 1 root disk 3, 16 Mar 22 06:12 /dev/hda16
brw-rw---- 1 root disk 3, 17 Mar 22 06:12 /dev/hda17
brw-rw---- 1 root disk 3, 18 Mar 22 06:12 /dev/hda18
brw-rw---- 1 root disk 3, 19 Mar 22 06:12 /dev/hda19
brw-rw---- 1 root disk 3, 2 Mar 22 06:12 /dev/hda2
brw-rw---- 1 root disk 3, 20 Mar 22 06:12 /dev/hda20
```

In this example, the major device number 3 is listed for all files. This refers to the driver for IDE hard drives on the first IDE channel. The minor device numbers for the first disk run from 1 to 15 (for SCSI hard drives) and up to 63 (for IDE hard drives) and refer to the various possible partitions.

Many device files are already available by default. Some of these, however, are never needed. If special device files are required for specific devices, these can be generated with the command **mknod**. The necessary parameters must be provided by the hardware manufacturer.

The null device /dev/null is also located in this directory. Program output that would normally be sent to the screen can be redirected to this device (for example, using redirects). The redirected data will be discarded.

The following are some important device files:

Table 5-2

Device	Device File	Description
Terminals	dev/console	The system console.
	/dev/tty1	The first virtual console, reachable by pressing **Ctrl + Alt + F1**.
Serial ports	/dev/ttyS0 /dev/ttyS*	The first serial port.
Parallel ports	/dev/lp0 /dev/lp*	The first parallel port.
Floppy disk drives	/dev/fd0 /dev/fd*	The first floppy disk drive. If the drives are addressed via the device files fd0 and fd1, the kernel tries to recognize the floppy disk format itself.
IDE hard drives	/dev/hda	The first IDE hard drive on the first IDE controller.
	/dev/hdc	The first IDE hard drive on the second IDE controller.
	/dev/hd*	To label the partitions, the device names are given numbers. Numbers 1 to 4 refer to the primary partitions, higher numbers to logical partitions. Example: **/dev/hda1** is the first primary partition on the first IDE hard drive.

(continued) **Table 5-2**

Device	Device File	Description
IDE CD-ROM drives	/dev/hd*	The drives are named in the same way as the IDE hard drives. This means that the CD-ROM drive **/dev/hdd** is the second drive on the second IDE controller.
SCSI hard drives	/dev/sda	The first SCSI hard drive.
	/dev/sda*	With SCSI hard drives, the device names are given numbers to label the various partitions. For example, **/dev/sda1** is the first primary partition on the first SCSI hard drive.
SCSI CD-ROM drives	/dev/scd0 /dev/scd*	The first SCSI CD-ROM drive.

Configuration Files (/etc/)

This directory and its subdirectories contain system configuration files. Almost all these files are ASCII files, which can be processed with any editor.

Normal users can read most of these files, but they cannot edit any of them. According to the FHS, no executable programs can be located here.

However, the subdirectories contain many shell scripts. Some important configuration files are listed in the following table:

Table 5-3

File	Description
/etc/SuSE-release	Version number of the installed SUSE LINUX Enterprise Server
/etc/inittab	Configuration file for the init process
/etc/init.d/*	Scripts for starting services
/etc/grub.conf	Configuration file of GRUB
/etc/modprobe.conf	Configuration file of the kernel modules
/etc/DIR_COLORS	Specifies the colors for ls
/etc/X11/ XF86Config	Configuration file of the X Window System
/etc/fstab	Table of the file systems automatically mounted at system start
/etc/profile	Startup file to configure the shell
/etc/passwd	User database; all information except passwords
/etc/shadow	Encrypted passwords of users
/etc/group	Database of user groups
/etc/cups/*	Files for the CUPS printing system
/etc/hosts	Allocation of computer names to IP addresses
/etc/motd	Welcome message after a user logs in (message of the day)
/etc/issue	Linux welcome message before the login prompt
/etc/sysconfig/*	System configuration files

Most installed services have at least one configuration file in the directory /etc/ or a subdirectory.

User Directories (/home/)

Every user on a Linux system has his own area in which to create and remove files. This area is called the home directory of the user. When a user logs in, she is in his own home directory.

Individual configuration files can be found in the user's home directory. These files have names that begin with a dot. These configuration files are hidden files, because files with names starting with a dot are normally not displayed by the command **ls**.

The following are the most important files in a user's home directory:

Table 5-4

File	Description
.profile	User's private login script
.bashrc	Configuration file for bash
.bash_history	List of commands previously run in bash

If there are no special settings, the home directories of all users are located beneath the directory /home/. The home directory of a user can also be addressed via the short cut "~", so **~/.bashrc** refers to the file .bashrc in the user's home directory.

In many cases, the directory /home/ is located on a different partition or can even be located on a different computer (with central administration of home directories).

Libraries (/lib/)

Many programs use specific functions that are also used by other programs. Such standard functions are removed from the actual program, stored in the system, and only called up when the program runs. They are called *shared libraries*.

The directory /lib/ contains the libraries that are used by programs in the directories /bin/ and /sbin/. The kernel modules (hardware drivers not compiled into the kernel) are located in the directory /lib/modules/.

You can find additional libraries below the directory /usr/.

Mount Points for Removable Media (/media/)*

SUSE LINUX creates directories such as the following in the directory /media/ (depending on your hardware) for mounting removable media:

- **/media/cdrom/.** Created for mounting CD-ROMs.
- **/media/cdrecorder/.** Created for mounting CDs in a CD burner.
- **/media/dvd/.** Created for mounting DVDs.
- **/media/floppy/.** Created for mounting floppy disks.

Application Directory (/opt/)

Installed programs can store their static files in the directory /opt/. First, a directory with the name of the application is created. The files are then stored in that directory.

Examples include GNOME (/opt/gnome/) and KDE3 (/opt/kde3/).

Home Directory of the Administrator (/root/)

The home directory of the system administrator is not located beneath /home/ like that of a normal user. Preferably, it should be on the same partition as the root directory, "/". Only then is it guaranteed that the user root can always log in without a problem and have her own configured environment available.

System Binaries (/sbin/)

The directory /sbin/ contains important programs for system administration. Programs that are run by normal users as well are located in /bin/.

Programs in the directory /sbin/ can also, as a rule, be run by normal users, but only to display the configured values. Changes to the configuration can only be made by the user root.

The following is an overview of important files in the directory /sbin/:

Table 5-5

File	Description
/sbin/SuSEconfig	Used to configure the overall system; evaluates entries in the configuration files in the directory /etc/sysconfig/ and writes further configuration files.
/sbin/conf.d/*	Contains more scripts from the SuSEconfig family. They are called up by /sbin/SuSEconfig.
/sbin/yast	Administration tool for SUSE LINUX Enterprise Server.
/sbin/fdisk	Modifies partitions.
/sbin/fsck	Checks file systems (file system check).
/sbin/init	Initializes the system.

(continued) **Table 5-5**

File	Description
/sbin/mkfs	Creates a file system (formatting).
/sbin/shutdown	Shuts down the system.

Data Directories for Services (/srv/)

The directory /srv/ contains subdirectories filled with data of various services. For example, the files of the Apache web server are located in the directory /srv/www/ and the FTP server files are located in the directory /srv/ftp/.

Temporary Area (/tmp/)

Various programs create temporary files that are stored in /tmp/ until they are deleted.

The Hierarchy below /usr/

The directory /usr/, in accordance with the FHS, represents a second hierarchical layer.

This is the location for all application programs, graphical interface files, additional libraries, locally installed programs, and commonly shared directories containing documentation.

These include the following:

Table 5-6

Directory	Description
/usr/X11R6/	Files of the X Window System
/usr/bin/	Almost all executable programs
/usr/lib/	Libraries

(continued) **Table 5-6**

Directory	Description
/usr/local/	Locally installed programs, now frequently found in the directory /opt/
/usr/sbin/	Programs for system administration
/usr/share/doc/	Documentation
/usr/share/man/	The manual pages (command descriptions)
/usr/src/	Source files of all programs and the kernel (if installed)

Variable Files (/var/)

The directory /var/ contains a hierarchy that is described in the FHS. This directory and its subdirectories contain files that can be modified while the system is running.

The following table provides an overview of the most important directories beneath /var/:

Table 5-7

Directory	Description
/var/lib/	Variable libraries (such as databases for the commands locate and rpm)
/var/log/	Log files for most services
/var/run/	Files with information on running processes
/var/spool/	Directory for queues (printers, email)
/var/lock/	Lock files to protect devices from multiple use

Process Files (/proc/)

Linux handles process information that is made available to users via the directory /proc/. This directory does not contain any real files and therefore does not occupy any space on the hard disk.

It is generated dynamically when it is accessed (for example, with **ls /proc/**). Each process has its own directory. The values in these directories can be read out as if they were in a file. Some values can also be set by writing to the corresponding "files." Changes to this virtual file system only have an effect as long as the system is running, however.

For example, the process **init** always has the process number "1." Information about it is therefore found in the directory /proc/1/. This directory contains the following files:

```
da10:~ # ls -l /proc/1
total 0
dr-xr-xr-x  3 root root 0 Apr 5 17:28 .
dr-xr-xr-x 62 root root 0 Mar 30 15:09 ..
dr-xr-xr-x  2 root root 0 Apr 5 17:36 attr
-r--------  1 root root 0 Apr 5 17:36 auxv
-r--r--r--  1 root root 0 Apr 5 17:28 cmdline
lrwxrwxrwx  1 root root 0 Apr 5 17:36 cwd -> /
-r--r--r--  1 root root 0 Apr 5 17:36 delay
-r--------  1 root root 0 Apr 5 17:36 environ
lrwxrwxrwx  1 root root 0 Apr 5 17:28 exe -> /sbin/init
dr-x------  2 root root 0 Apr 5 17:36 fd
-rw-------  1 root root 0 Apr 5 17:36 map_base
-r--r--r--  1 root root 0 Apr 5 17:36 maps
-rw-------  1 root root 0 Apr 5 17:36 mem
-r--r--r--  1 root root 0 Apr 5 17:36 mounts
lrwxrwxrwx  1 root root 0 Apr 5 17:36 root -> /
-r--r--r--  1 root root 0 Apr 5 17:28 stat
-r--r--r--  1 root root 0 Apr 5 17:36 statm
-r--r--r--  1 root root 0 Apr 5 17:36 status
dr-xr-xr-x  3 root root 0 Apr 5 17:36 task
-r--r--r--  1 root root 0 Apr 5 17:36 wchan
```

The contents of the files can be viewed with the command **cat**, which shows the status of the process, shown in the following:

```
da10:~ # cat /proc/1/status
Name:  init
State: S (sleeping)
SleepAVG:    26%
Tgid:  1
Pid:  1
PPid:  0
TracerPid:   0
Uid:  0    0    0    0
Gid:  0    0    0    0
FDSize: 32
Groups:
VmSize:   588 kB
VmLck:     0 kB
VmRSS:    108 kB
VmData:   136 kB
VmStk:     8 kB
VmExe:    432 kB
VmLib:     0 kB
Threads:    1
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: fffffffd770d8fc
SigCgt: 00000000288b2603
CapInh: 0000000000000000
CapPrm: 00000000ffffffff
CapEff: 00000000fffffeff
da10:~ #
```

In this example, a list is displayed of what the process is called (**init**), what state it is in (**sleeping**), and to which user it belongs (**Uid: 0** for root).

In addition to directories for each individual process, /proc/ also includes directories and files containing information about the state of the system.

The following are the most important files and directories:

Table 5-8

File	Description
/proc/cpuinfo	Information about the processor
/proc/dma	Use of the DMA ports (Direct Memory Access)
/proc/interrupts	Use of the interrupts
/proc/ioports	Use of the intrasystem I/O ports
/proc/filesystems	File system formats that the kernel understands
/proc/modules	Active modules
/proc/mounts	Mounted file systems
/proc/net/*	Network-specific information and statistics in human-readable form, for example, **ip_fwchains**, **ip_fwnames**, and **ip_masquerade** (IP firewall chains for kernels older than 2.4, meanwhile replaced by iptables for kernel 2.4 and 2.6)
/proc/partitions	Existing partitions
/proc/pci	Existing PCI devices
/proc/scsi/	Connected SCSI devices
/proc/sys/*	System and kernel information
/proc/version	Kernel version

System Information Directory (/sys/)

The directory /sys/ provides information in the form of a tree structure on various hardware buses, hardware devices, active devices, and their drivers.

Mount Point for Temporarily Mounted File Systems (/mnt/)

The standard directory for integrating file systems is /mnt/. It should only be used for temporary purposes. For permanent mounts, you should create an appropriately named directory.

In the following example, the hard drive partition /dev/hda7 is mounted at the position /mnt/ in the directory tree using the command mount:

```
da10:~ # mount /dev/hda7 /mnt
```

All files on this partition can now be reached via the directory /mnt/. To remove this partition again, you use the command umount:

```
da10:~ # umount /mnt
```

If you do not include any options with the command mount, the program tries out several file system formats. If you want to specify a specific file system, use the option -t.

If the file system format is not supported by the kernel, the command is aborted, and you receive an error message. In this case, you must compile a new kernel that supports the file system format.

Directories for Mounting Other File Systems

Other file systems such as other hard drive partitions, directories from other computers on the network, or removable media (floppy disk, CD-ROM, or removable hard drive) can be mounted to the file system at any point.

A directory must exist at the point where you intend to mount the file system. This directory is referred to as the *mount point*. Once mounted, the complete directory structure of the mounted file system can be found beneath this directory.

In most cases, only the user root can mount and unmount directories. Removable media, such as floppy disks and CDs, can be changed by a normal user.

To mount a file system, enter the command **mount**, specifying the device file and the directory to which the file system should be mounted.

A file system can be removed again with the command **umount**. The file /etc/mtab, which is updated by the command mount, shows which file systems are currently mounted.

You can mount file systems to directories that are not empty. The existing contents of these directories, however, will no longer be accessible. After the file system is removed, this data will become available again.

Because the mounted file system does not have to be on a local hard disk, directories can be shared with many computers. This approach is often used for the home directories of users, which are located centrally on one machine and exported to other computers in the network.

The following directories cannot be imported from other machines; they must always be located locally on each computer:

Table 5-9

Directory	Description
/bin/	Important programs
/boot/	Kernel and boot files
/dev/	Device files
/etc/	Configuration files

(continued) **Table 5-9**

Directory	Description
/lib/	Libraries
/sbin/	Important programs for system administration

The following are some of the directories that can be shared:

Table 5-10

Directory	Description
/home/	Home directories
/opt/	Applications
/usr/	The hierarchy below /usr/

Objective 2 Identify File Types in the Linux System

The Linux file system is distinct from the file systems of other operating systems because of the various file types.

The file types in Linux referred to as normal files and directories are also familiar to other operating systems:

- Normal Files
- Directories

Additional types of files are UNIX-specific:

- Device Files
- Links
- Sockets
- FIFOs

Normal Files

Normal files refer to files as they are also known to other operating systems: a set of contiguous data addressed with one name. This includes all the files normally expected under this term (such as ASCII texts, executable programs, or graphics files).

You can use any names you want for these files—there is no division into filename and file type (such as report.txt).

A number of filenames still retain this structure, but these are requirements of the corresponding applications, such as a word processing program or a compiler.

Directories

Directories contain two entries with which the structure of the hierarchical file system is implemented.

One of these entries (".") points to the directory itself. The other entry ("..") points to the entry one level higher in the hierarchy.

Device Files

Each piece of hardware (with the exception of network cards) in a Linux system is represented by a *device file*. These files represent links between the hardware components or the device drivers in the kernel and the applications.

Every program that wants to access hardware must access it through the corresponding device file. The programs write to or read from a device file. The kernel then ensures that the data finds its way to the hardware or can be read from the file.

Links

Links are references to files located at other points in the file system. Data maintenance is simplified through the use of such links. Changes only need to be made to the original file. The changes are then automatically valid for all links.

Sockets

A *socket* refers to a special file with which data exchange between two locally running processes can be implemented through the file system.

FIFOs

FIFO (first in first out) or *named pipe* is a term used for files used to exchange data between processes. However, the file can exchange data in one direction only.

Objective 3 Change Directories and List Directory Contents

The prompt of a shell terminal contains the current directory (such as **tux@da10:~**). The tilde (~) indicates that you are in the user's home directory.

You can use the following commands to list the contents of a directory, change directory and display the current directory:

- ls
- cd
- pwd

ls

The command **ls** (list) lists the specified files. If a directory is included with ls, the directory's contents are displayed. Without an option, the contents of the current directory are listed.

The following are the most important options you can use with **ls**:

Table 5-11

Option	Meaning
(none)	Displays the contents of the current directory in several columns (file and directory names only).
-a	Also displays hidden files (such as **.bashrc**)
-F	After each name, a character indicates the file type. (/ for directories, * for executable files, | for FIFO files, @ for symbolic links). This is helpful when the terminal does not display colors.
-l	(long list) Gives a detailed list of all files. For each filename, information about permissions, modification time, and size is included.

(continued) **Table 5-11**

Option	Meaning
-t	Files are sorted by date of modification. Combined with the option -r, the output is in reverse order (the newest file is displayed last).
-R	Output is recursive, including all subdirectories.
-u	Sorted by date of last access.

The following is an example of using ls:

```
tux@da10:/ > ls var/
adm cache games lib lock log mail opt run spool tmp X11R6 yp
tux@da10:/ > ls -l var/
total 2
drwxr-xr-x 10 root root 272 2004-03-29 12:31 adm
drwxr-xr-x  6 root root 144 2004-03-29 11:58 cache
drwxrwxr-x  2 games games 48 2004-03-23 18:41 games
drwxr-xr-x 22 root root 624 2004-04-13 04:17 lib
drwxrwxr-x  4 root uucp  96 2004-04-06 14:45 lock
drwxr-xr-x  6 root root 800 2004-04-05 17:29 log
lrwxrwxrwx  1 root root  10 2004-03-29 11:23 mail -> spool/mail
drwxr-xr-x  3 root root  72 2004-03-29 11:26 opt
drwxr-xr-x 12 root root 776 2004-04-08 11:21 run
drwxr-xr-x 11 root root 296 2004-03-29 12:00 spool
drwxrwxrwt  5 root root 144 2004-04-06 14:44 tmp
drwxr-xr-x  4 root root 120 2004-03-29 11:47 X11R6
drwxr-xr-x  3 root root 104 2004-03-29 11:46 yp
tux@da10:/ >
```

Each line shows the file permissions (covered in "Manage File Permissions and Ownership" on 7-28), the number of hard links to the file (covered in "Link Files" on 5-49), owner and owning group, file size, time of last modification of content, and the filename.

cd

You can use the command **cd** (change directory) to change between directories. Some examples include the following:

Table 5-12

Command	Meaning
cd plan	Change to the subdirectory plan.
cd /etc	Change directly to the directory /etc/ (absolute path).
cd	Change from any directory to the home directory.
cd ..	Move one directory level higher.
cd ../..	Move two directory levels higher.
cd -	Move to the last valid directory.

pwd

You can use the command **pwd** (print working directory) to display the path of the current directory. If you enter pwd with the -P option, pwd prints the physical directory without any symbolic links, as shown in the following:

```
tux@da10:~ > ls -l doc/
lrwxrwxrwx  1 tux users  15 2004-02-12 08:43 doc -> /usr/share/doc/
tux@da10:~ > cd doc/
tux@da10:~ > pwd
/home/tux/doc
tux@da10:~ > pwd -P
/usr/share/doc
tux@da10:~ >
```

Exercise 5-2 Change Directories and List Directory Contents

Smooth administration of a Linux system requires familiarity with the directory tree and how to move within it. The purpose of this exercise is to show you how to orient yourself and move about within that tree.

To change the current directory and list the directory contents, do the following:

1. Describe what directories the following characters refer to:
   - ❑ **/**
   - ❑ **~**
   - ❑ **.**
   - ❑ **..**
2. From the KDE desktop, open a terminal window (Konsole).
3. Change to the directory /tmp/ by entering cd **/tmp**.
4. Change to the home directory by entering **cd ~**.
5. Display the name of the current directory by entering **pwd**.
6. Change to the directory /usr/share/doc by entering

   cd /usr/share/doc
7. Display the name of the current directory by entering **pwd**.
8. Change back to the last directory (home) by entering **cd -**.
9. Display the name of the current directory by entering **pwd**.
10. Display the content of the current directory by entering **ls**.
11. Display the content of the current directory including the hidden files by entering **ls -a**.
12. Display the permissions and the file size of the directories starting with "D" in the current directory by entering

    ls -ld D*

13. View the permissions and the file size of all the files in the current directory by entering **ls -la**.
14. Close the terminal window by entering **exit**.

(End of Exercise)

Objective 4 Create and View Files

To create and view files, you need to know how to do the following:

- Create a New File with touch
- View a File with cat
- View a File with less
- View a File with head and tail

Create a New File with touch

You can use the command **touch** to change the time stamp of a file or create a new file with a size of 0 bytes. The following are the most important touch options:

Table 5-13

Command	Description
-a	Changes only the time of the last read access (**access time**).
-m	Changes only the time of the last modification of file content (**modification time**).
-r *file*	Sets the time stamp of *file* instead of the current time.
-t *time*	Instead of the current time, sets *time.* (structure: [[CC]YY]MMDDhhmm.[ss] ([Century]Year] Month Day Hour Minute [Seconds], (use two digits for each variable)).

The following is an example of using touch:

```
tux@da10:~> ls
bin Desktop Documents public_html
tux@da10:~> touch example
tux@da10:~> ls
bin Desktop Documents example public_html
tux@da10:~>
```

View a File with cat

You can use the command **cat** to view the contents of a file (comparable to the command "type" in DOS). The command must include the filename of the file you want to see, as shown in the following:

```
tux@da10:~> cat /etc/HOSTNAME
da10.digitalairlines.com
tux@da10:~>
```

View a File with less

You can use the command **less** to displays the contents of a file page by page. Even compressed files (such as .gz and .bz2) can be displayed. You can use the following keystrokes with less:

Table 5-14

Keystroke	Description
Spacebar	Move one screen down.
b	Move one screen up.
Down-arrow	Move one line down.
Up-arrow	Move one line up.
/pattern	Search forward for a pattern from current cursor position.

(continued) **Table 5-14**

Keystroke	Description
?*pattern*	Search backward for a pattern from the current cursor position.
n	Move to the next instance in the search for the pattern.
N	Move to the previous instance in the search for the pattern.

View a File with head and tail

With the command **head**, you can view only the first few lines of a file. The command **tail** shows you only the last few lines of a file.

By default, these commands only show ten lines. To change this number, just append the option ***-number***, like **-20**.

When used with the command tail, the option **-f** displays a continuously updated view of the last lines of a file. If a line is added at the end of the file while **tail -f** is running, the line is displayed. This is a very useful feature for observing log files.

The following is an example of using the command head:

```
tux@da10:~> head /usr/share/doc/release-notes/RELEASE-NOTES.en.html
<H1>Release Notes for SUSE LINUX Enterprise Server 9 for x86</H1>

<P>
These release notes cover the following areas:
<ul>
<li>General: Information that everybody should read.</li>
<li>Update: Explains changes that are not mentioned in the Admin Guide,
chapter 2.</li>
<li>Installation: Additional pertinent information for the
Installation.</li>
<li>Updates and Features: This sections contains a number of technical
changes and enhancements for the experienced user.</li>
<li>Providing Feedback</li>
tux@da10:~> tail -5 /usr/share/doc/release-notes/RELEASE-NOTES.en.html
http://www.suse.com/feedback.
</P>
<P>
Your SUSE LINUX Enterprise Team
</P>
tux@da10:~>
```

Exercise 5-3 Create and View Files

To be able to view configuration and log files is a necessary part of system administration. Various tools exist for this purpose, and you choose the appropriate one depending on whether you want to view the complete file or only part of it.

To create and view files or parts of them, do the following:

1. Open a terminal window, log in as root (**su -**) with a password of **novell**.
2. Create a new, empty file by entering

   touch new_file
3. Display the contents of the file /var/log/messages by entering

   cat /var/log/messages
4. Display the contents of /var/log/messages page by page by entering

   less /var/log/messages
5. Find the first occurrence of the word "root" by entering **/root**.
6. Find the next occurrence of the word "root" by typing **n**.
7. Navigate through the output by using the cursor keys and the **Page Up** and the **Page Down** keys.
8. Quit the display and return to the command line by typing **q**.
9. Display the first five lines of the file /var/log/messages by entering

   head -n 5 /var/log/messages
10. View a continuously-updated display of the last lines of the file /var/log/messages by entering

    tail -f /var/log/messages
11. Open a second terminal window.

12. Arrange the terminal windows on the desktop so that you can see the content of both.

13. In the second terminal window, log in as root (**su -**); then enter an invalid password (such as **suse**).

    Notice that the failed login attempt is logged in the first terminal window.

14. In the second terminal window, log in as root (**su -**) with a password of **novell**.

    The login is logged in the first terminal window.

15. Log out as root in the second terminal window by entering **exit**.

16. Close the second terminal window by entering **exit**.

17. Stop the tail process in the first terminal window by pressing **Ctrl + c**.

18. Log out as root by entering **exit**.

19. Close the terminal window.

(End of Exercise)

Objective 5 Manage Files and Directories

In this objective, you learn how to:

- Copy and Move Files and Directories
- Create Directories
- Delete Files and Directories
- Link Files

Copy and Move Files and Directories

To copy and move files and directories, you need to know how to:

- Move Files with mv
- Copy Files with cp

Move Files with mv

You can use the command **mv** (move) to move one or more files to another directory, as in the following:

```
mv *.txt /tmp
```

You can also use the command mv to rename a file, as in the following:

```
mv recipe new_recipe
```

The following are two options you can use with mv:

Table 5-15

Option	Description
-i	Asks for confirmation before moving or renaming a file. This prevents existing files with the same name from being overwritten.

(continued) **Table 5-15**

Option	Description
-u	Only moves files that are newer than the target files of the same name.

Copy Files with cp

You can copy files and directories with the command **cp** (copy). The syntax for using cp is

cp *source destination*

When using the command cp, you need to remember the following:

- The command cp overwrites existing files without confirmation.
- You can avoid automatic overwriting by using the option -i. This option requires confirmation before overwriting occurs.
- If you want to copy just the contents of a directory (without the directory itself), the target directory must already exist. An example is making a backup copy of a directory using a directory with a different name.

  For example, to copy the directory /tmp/quarterly-1/ with all its subdirectories to the directory /tmp/expenses/ (which already exists), you would enter the following:

  cp -R /tmp/quarterly-1 /tmp/expenses

  The result is a directory /tmp/expenses/quarterly-1/.

To copy the contents of the directory proposals/ and all its files, including hidden files and subdirectories, to the existing directory proposals_old/, do the following:

```
tux@da10:~ > ls -a proposals
. .. .hidden quarterly-1 quarterly-2 quarterly-3 quarterly-4
tux@da10:~ > cp -r proposals/. proposals_old
tux@da10:~ > ls -a proposals_old
. .. .hidden quarterly-1 quarterly-2 quarterly-3 quarterly-4
```

To avoid copying the hidden files, do the following:

```
tux@da10:~ > cp -r proposals/* proposals_old
tux@da10:~ > ls -a proposals_old
. ..  quarterly-1 quarterly-2 quarterly-3 quarterly-4
```

You can use the following options with cp:

Table 5-16

Option	Description
-a, --archive	Copies a directory and subdirectories (compare with -R); symbolic links, file permissions, owners, and time stamps are not changed
--help	Displays the options of cp
-i, --interactive	Asks before overwriting
-l, --link	Links files instead of copying them
-R, -r, --recursive	Copies directories recursively (the directory and any subdirectories)
-s, --symbolic-link	Makes symbolic links instead of copying files
-u, --update	Copies a file only when the source file is newer than the destination file or when the destination file is missing

Exercise 5-4 Copy and Move a File and a Directory

Copying, moving and renaming files are basic and frequent operations done with files. Most probably you are already very familiar with these operations on a graphical desktop environment.

The purpose of this exercise is to get used to performing these operations on the command line.

To copy and move files and directories, do the following:

1. Open a terminal window and log in as root (**su -**) with a password of **novell**.
2. Rename new_file to my_file by entering

   mv new_file my_file
3. Verify that the file was renamed by entering **ls -l**.
4. Copy my_file by entering **cp my_file my_file1**.
5. Verify that my_file1 was created by entering **ls -l my***.
6. Copy the files /usr/bin/rename and /usr/bin/mcopy to the directory /tmp/ by entering

   cp /usr/bin/rename /usr/bin/mcopy /tmp
7. Verify that the files were copied by entering

   ls -l /tmp
8. Move the file /tmp/mcopy to the home directory by entering

   mv /tmp/mcopy ~
9. Verify the move by entering **ls -l**.
10. Move and rename the file /tmp/rename to ~/my_file2 by entering

    mv /tmp/rename ~/my_file2
11. Verify that the file my_file2 exists by entering **ls -l**.
12. Copy the complete directory /bin/ to the home directory with the new directory named my_dir by entering

```
cp -r /bin /home/my_dir
```

13. Verify that the files were copied by entering

```
ls -l /home/my_dir
```

14. Close the terminal window.

(End of Exercise)

Create Directories

You can use the command **mkdir** (make directory) to create new directories (such as **mkdir proposal**). Use the option -p to create a complete path, as shown in the following:

```
mkdir -p proposal/january
```

Exercise 5-5 Create Directories

The purpose of this exercise is to show you how to create directories.

To create directories, do the following:

1. Open a terminal window and su to root (**su -**) with a password of **novell**.
2. Create a directory named new_dir inside the directory my_dir by entering

   ```
   mkdir  /home/my_dir/new_dir
   ```

3. Verify that the directory was created by entering

   ```
   ls  /home/my_dir
   ```

4. Create a directory new_dir that includes a new directory empty_dir by entering

   ```
   mkdir  -p  ~/new_dir/empty_dir
   ```

5. Verify that new_dir was created by entering **ls**.
6. Verify that empty_dir was created by entering

   ```
   ls  new_dir
   ```

7. Close the terminal window.

(End of Exercise)

Delete Files and Directories

In this section you learn how to:

- Use rmdir to Delete Empty Directories
- Use rm to Delete Files and Directories

Use rmdir to Delete Empty Directories

You can use the **rmdir** (remove directory) command to remove the indicated directory or directories (for example, **rmdir proposal**). The directory or directories must be empty before you can delete them.

Use rm to Delete Files and Directories

You can use the command **rm** (remove) to delete files, as shown in the following:

rm part*

The command in this example deletes all files in the current directory that begin with **part** without asking for confirmation. If the user does not have sufficient permissions to delete a file, this file is ignored.

To delete directories, use the option **-r**, as shown in the following example:

```
tux@da10:~ > mkdir -p testdir/subdir
tux@da10:~ > rm testdir
rm: cannot remove `testdir': Is a directory
tux@da10:~ > rmdir testdir
rmdir: `testdir': Directory not empty
tux@da10:~ > rm -r testdir
tux@da10:~ >
```

Files deleted with the command rm cannot be restored.

You can use the following two options with rm:

Table 5-17

Option	Description
-i	Asks for confirmation before deleting
-r	Allows directories and their contents to be deleted

Exercise 5-6 Delete Files and Directories

The purpose of this exercise is to show you how to delete files and directories on the command line.

When deleting files and directories it is especially important to work only with the permissions necessary for the task and not as root, if possible. As root, you can easily destroy your installation with the rm command, and there is no undelete in Linux.

Think twice before hitting enter on an rm command issued as root!

To delete files and directories, do the following:

1. Open a terminal window and su to root (**su -**) with a password of **novell**.
2. Try to remove the directory ~/new_dir by entering

   rmdi new_dir

   A message is displayed indicating that the directory cannot be removed. This is because the directory is not empty.
3. Remove the directory ~/new_dir/empty_dir by entering

   rmdir new_dir/empty_dir
4. Verify that the directory /empty_dir has been removed by entering

   ls new_dir
5. Remove the directory ~/new_dir by entering

   rmdir new_dir
6. Verify that the directory was removed by entering **ls**.
7. Remove the file /home/my_dir/login by entering

   rm /home/my_dir/login
8. Verify that the file has been removed by entering

   ls /home/my_dir/login

9. Remove all files with names that begin with "a" in the directory /home/my_dir/ by entering

   rm -i /home/my_dir/a*

10. Confirm every warning by entering **y**.

11. Remove the directory /home/my_dir/ and its content by entering

    rm -r /home/my_dir

12. Verify that the directory has been removed by entering

    ls /home/

13. Close the terminal window.

(End of Exercise)

Link Files

File system formats in Linux keep data and administration information separate. How data organization takes place differs from one file system format to the next.

Each file is described by an inode (index node or information node). To see the inode number you can enter **ls -i**.

Each inode has a size of 128bytes and contains all the information about this file besides the filename. This includes information such as owner, access permissions, size, time details (such as time of modification of content of file, time of access, and time of modification of the inode), and pointers to the data blocks of this file.

The command **ln** creates a link. A *link* is a reference to a file. You can use a link, to access a file from anywhere in the file system using different names for it. The file itself exists only once on the system, but it can be found under different names.

Linux recognizes two kinds of links: hard links and symbolic links (also called *soft links*).

You create a *hard link* by using the command ln, which points to the inode of an already existing file. Thereafter, the file can be accessed under both names—that of the file and that of the link, and you can no longer discern which name existed first or how the original file and the link differ.

The following is an example of using the command ln:

```
tux@da10:~/sell > ls -li
total 4
88658 -rw-r--r-- 1 tux users 82 2004-04-06 14:21 old
tux@da10:~/sell > ln old new
tux@da10:~/sell > ls -li
total 8
88658 -rw-r--r-- 2 tux users 82 2004-04-06 14:21 old
88658 -rw-r--r-- 2 tux users 82 2004-04-06 14:21 new
tux@da10:~/sell >
```

The link counter (shown after the file permissions) has changed from 1 to 2 because of the additional hard link to the file.

Hard links can only be used when both the file and the link are in the same file system, because inode numbers are only unique within the same file system.

You can create a *symbolic link* with the command ln and the option -s. A symbolic link is assigned its own inode—the link refers to a file, so a distinction can always be made between the link and the actual file.

The following is an example of creating a symbolic link:

```
tux@da10:~/sell > ls -li
total 4
88658 -rw-r--r-- 1 tux users 82 2004-04-06 14:21 old
tux@da10:~/sell > ln -s old new
tux@da10:~/sell > ls -li
total 4
88658 -rw-r--r-- 1 tux users 82 2004-04-06 14:21 old
88657 lrwxrwxrwx 1 tux users 3 2004-04-06 14:27 new -> old
tux@da10:~/sell >
```

The link counter (shown after the file permissions) has not changed, because only hard links are counted here.

If you use symbolic links, you are not limited by the boundaries of the file system, because the name of the object is shown, not the object itself. The disadvantage is that a symbolic link can point to a non-existing object if the object and its corresponding name no longer exist.

For e ample, if you erase the file **old** in the preceding *example,* in SLES 9 **new** will be shown in a different color in the output of ls, indicating that it points to a non-existent file:

```
tux@da10:~/sell > rm old
tux@da10:~/sell > ls -li
total 0
88657 lrwxrwxrwx 1 tux users 3 2004-04-06 14:27 new -> old
tux@da10:~/sell >
```

Another advantage of symbolic links is that you can create links to directories.

Exercise 5-7 Link Files

Links are very convenient in administration, as they help to avoid having different versions of the same file within the file system.

The purpose of this exercise is to learn how to set hard and symbolic links and know the difference between those two.

To link a file, do the following:

1. Open a terminal window and su to root (**su -**) with a password of **novell**.
2. Enter the following to create a symbolic link to the howto index file in the directory /usr/share/doc/howto/en/html:

   ln -s /usr/share/doc/howto/en/html/index.html howto-sym
3. Enter the following to create a hard link to the howto index file in the directory /usr/share/doc/howto/en/html:

   ln /usr/share/doc/howto/en/html/index.html howto-hard
4. Display the links by entering **ls -l**.

   The symbolic link identifies the file it is linked to.
5. View the contents of the file /usr/share/doc/howto/en/html/index.html by entering

   cat howto-hard
6. Create a file named **hello** by entering

   echo hello > hello
7. Create a symbolic link to the file by entering

   ln -s hello hello-symlink
8. View the content of the file hello via its symbolic link by entering

   cat hello-symlink
9. Delete the original file by entering **rm hello**.

10. View the link by entering

    ```
    ls  -l  hello-symlink
    ```

    Note that the link is now highlighted to indicate that it points to a non-existant file.

11. Delete the link by entering

    ```
    rm  hello-symlink
    ```

12. Close the terminal window.

(End of Exercise)

Objective 6 Find Files

In this objective you learn how to find files and programs using the following commands:

- KFind
- find
- locate
- whereis
- which
- type command

KFind

Sometimes you need to find a file so you can edit it, but you do not know exactly where it is located in the file system. You might know the name of this file or only a part of the name.

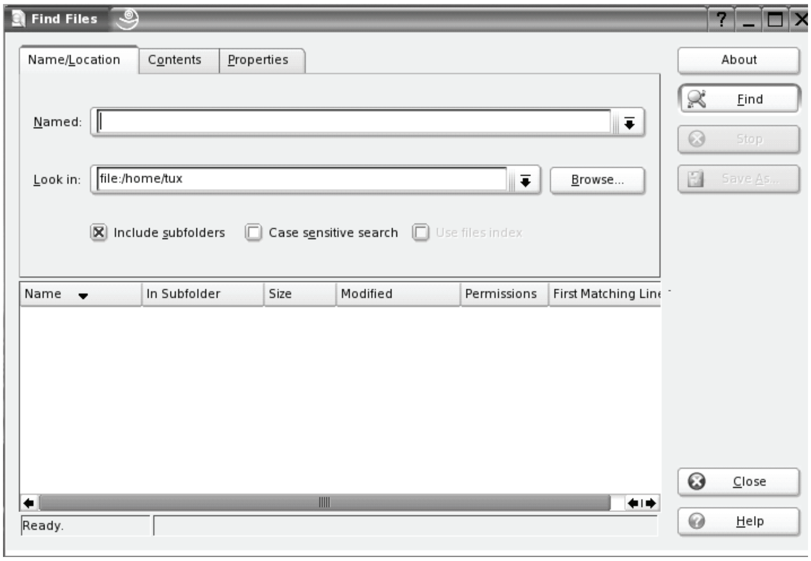
At another time, you might need a list of all files that have been modified in the last two days or that exceed a certain size.

You can use the KFind program to find files with specific features. You can start KFind from the KDE menu with the **Find Files** entry.

You can also start KFind directly in Konqueror by selecting **Tools > Find File**.

The following shows the Kfind dialog:

Figure 5-3



Look in: file:/home/tux

[✓ Include subfolders] Case sensitive search Use files index

In the Named field, enter the name of the file you want to find. If you don't know the entire name of the file, you can use the wildcards “?” for one character and “*” for none, one, or several characters.

For example, suppose the following files exist:

- File
- file
- File1
- File1a
- File1b
- File2

- File2a
- MyFile

The following table shows the results of three different search strings:

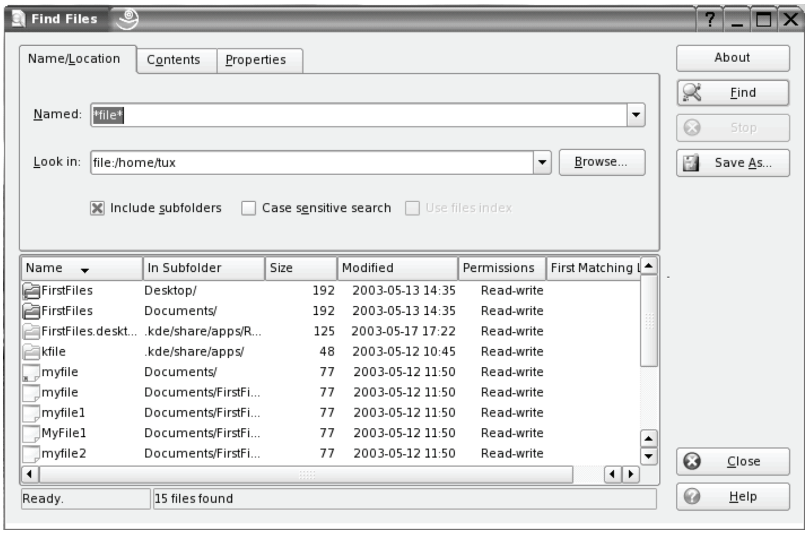
Table 5-18

Search String	Files Found
File?	File1
	File2
File*	File
	File1
	File1a
	File1b
	File2
	File2a
?ile*	File
	file
	File1
	File1a
	File1b
	File2
	File2a

Enter the directory you want to search in the Look In field. If you want to include all subdirectories in the search, select **Include Subfolders**.

Select the **Find** button to start the search process. All matching files and directories are shown in the lower window with details of their locations, as in the following:

Figure 5-4



Named: *file*

Look in: file:/home/tux

[✓ Include subfolders] Case sensitive search Use files index

Name	In Subfolder	Size	Modified	Permissions	First Matching L
FirstFiles	Desktop/	192	2003-05-13 14:35	Read-write	
FirstFiles	Documents/	192	2003-05-13 14:35	Read-write	
FirstFiles.deskt...	.kde/share/apps/R...	125	2003-05-17 17:22	Read-write	
kfile	.kde/share/apps/	48	2003-05-12 10:45	Read-write	
myfile	Documents/	77	2003-05-12 11:50	Read-write	
myfile	Documents/FirstFi...	77	2003-05-12 11:50	Read-write	
myfile1	Documents/FirstFi...	77	2003-05-12 11:50	Read-write	
MyFile1	Documents/FirstFi...	77	2003-05-12 11:50	Read-write	
myfile2	Documents/FirstFi...	77	2003-05-12 11:50	Read-write	

Ready. 15 files found

You can narrow the search by specifying certain criteria in Contents and Properties. In Contents, you can specify the file type, a string contained in the file, or the file size. In Properties, you can specify the date when the file to find was created or modified.

find

To search for files on the command line, you can use the command **find**. The following is the syntax for the command find:

find path *criteria action*

This command has many options, a few of which are explained here. You can use the following arguments with the command:

- ***path*.** The section of the file system to search (the specified directory and all its subdirectories). If nothing is specified, the file system below the current directory is used.
- ***criteria*.** The properties the file should have (refer to the following):

Table 5-19

Option	Description
--mtime *number*	Searches for files whose content changed no earlier than a specified *number* x 24 hours ago.
--gid *number*	Searches for files with the numeric GID (Group ID) *number*.
--group *name*	Searches for files that are owned by the group *name*. Instead of a name, the numeric GID is allowed.
--name *pattern*	Searches for files whose names contain the given *pattern*. If the pattern contains meta characters or wildcards, it must be enclosed by quotation marks. Otherwise the wildcards will be interpreted by the shell and not by find.
--size [+/-]*size*	Matches files that are above or below a certain *size*. As an argument, the size (in blocks of 512 bytes) is given. The suffix "c" switches to byte and "k" to blocks of 1024 bytes. A preceding "+" stands for all larger files and a "-" for all smaller files.
--type *file_type*	Searches for a *file type*. A file type can be one of the following: "d" for a directory, "f" for a file, or "l" for a symbolic link.

(continued) **Table 5-19**

Option	Description
--uid *number*	Searches for files with the numeric UID (User ID) *number*.
--user *name*	Searches for files, which are owned by user *name*. Instead of a name, the numeric UID is allowed.

Examples:

To find files with a certain extension beneath the /usr/share/doc/ directory, enter the following:

```
tux@da10:~ > find /usr/share/doc/ -name "*.txt" -type f
/usr/share/doc/xine/faq/faq.txt
...
```

To find files that were changed within the last 24 hours, enter

```
tux@da10:~ > find ~ -mtime 0
.
./.xsession-errors
...
```

- ***action.*** Options that influence the following conditions or control the search as a whole, such as the following:
  - --print
  - --exec *command*

You can use the option -exec to call up another command. This option is frequently used to link **find** and **grep**, as shown in the following:

```
tux@da10:~ > find ~ -name 'letter*' -type f -exec grep appointment {} \; \
-print
appointment for next meeting: 23.08.
/home/tux/letters/letter_Meier
tux@da10:~ >
```

In this example, the command find searches for files containing letters in their names, and then passes the names of the files found with -exec to the following command (in this case, **grep appointment {}**).

The two brackets {} are placeholders for the filenames that are found and passed to the command grep. The semicolon closes the -exec instruction. Because this is a special character, it is masked by placing a backslash in front of it.

When grep is used alone, it searches for a specific expression in a file whose exact position in the file system is known. When used in combination with find, the search is for a file that contains a certain expression but whose location is unknown.

locate

The command **locate** is an alternative to **find -name** (the package findutils-locate must be installed). The command find must search through the selected part of the file system, a process that can be quite slow.

On the other hand, locate searches through a database previously created for this purpose (/var/lib/locatedb), making it much faster.

The database is automatically created and updated daily by SLES 9. But changes made after the update has been performed are not taken into account by locate, unless the database is updated manually using the command **updatedb**.

The following example shows the output of locate:

```
tux@da10:~ > locate letter_Miller
/home/tux/letters/letter_Miller
```

The following example shows that a search with locate returns all files whose names contain the search string:

```
tux@da10:~ > locate umount
/bin/umount
/lib/klibc/bin/umount
/opt/kde3/share/icons/crystalsvg/scalable/devices/3floppy_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/5floppy_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/camera_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/cdaudio_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/cdrom_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/cdwriter_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/dvd_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/hdd_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/mo_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/nfs_umount.svgz
/opt/kde3/share/icons/crystalsvg/scalable/devices/zip_umount.svgz
/usr/bin/humount
/usr/bin/smbumount
/usr/share/man/man1/humount.1.gz
/usr/share/man/man2/umount.2.gz
/usr/share/man/man2/umount2.2.gz
/usr/share/man/man8/smbumount.8.gz
/usr/share/man/man8/umount.8.gz
tux@da10:~ >
```

To learn more about locate, enter **man locate**.

whereis

The command **whereis** returns the binaries (option **-b**), manual pages (option **-m**), and the source code (option **-s**) of the specified command.

If no option is used, all this information is returned, if the information is available. This command is faster than find, but it is less thorough.

The following is an example of using whereis:

```
tux@da10:~ > whereis grep
grep: /bin/grep /usr/bin/grep
/usr/share/man/man1/grep.1.gz
/usr/share/man/man1p/grep.1p.gz
tux@da10:~ > whereis -b grep
grep: /bin/grep /usr/bin/grep
tux@da10:~ > whereis -m grep
grep: /usr/share/man/man1/grep.1.gz
/usr/share/man/man1p/grep.1p.gz
tux@da10:~ > whereis -s grep
grep:
tux@da10:~ >
```

For more information about whereis, enter **man whereis**.

which

The command **which** searches all paths listed in the variable PATH for the specified command and returns the full path of the command.

This command is especially useful if several versions of a command exist in different directories and you want to know which version is executed when entered without specifying a path.

The following is an example of using the command which:

```
tux@da10:~ > which find
/usr/bin/find
tux@da10:~ > which cp
/bin/cp
tux@da10:~ > which grep
/usr/bin/grep
tux@da10:~ >
```

For more information on which, enter **man which**.

type command

The command **type *command*** can be used to find out what kind of command is executed when command is entered—a shell built-in command or an external command. The option **-a** delivers all instances of a command bearing this name in the file system.

The following is an example of using the command type:

```
tux@da10:~ > type type
type is a shell built in
tux@da10:~ > type grep
grep is /usr/bin/grep
tux@da10:~ > type -a grep
grep is /usr/bin/grep
grep is /bin/grep
tux@da10:~ >
```

Exercise 5-8 Find Files

With around 100,000 files in a usual installation it is essential to be able to find files effectively within the file system. It is possible to search using part of the file name, but also to look for files that contain a certain string, as covered in the exercise after this one.

To find files and directories, do the following:

1. Open a terminal window (you do not need to su to root).
2. Find the type of the command ll by entering **type ll**.
3. Find the manual pages of the command find by entering

   whereis -m find
4. Find the path of the program KFind by entering

   which kfind
5. Start KFind by entering **kfind &**.

   A Find Files dialog appears on the KDE desktop.
6. Find all files in the home directory whose names start with "my" by entering **my*** in the Named field; then select **Find**.
7. Find all files in the directory /bin/ whose names consist of two characters by entering **??** in the Named field and **/bin** in the Look in field; then select **Find**.
8. Find all files in the directory /tmp/ that were changed or created in the last 24 hours by doing the following:

   a. Enter ***** in the Named field and **/tmp** in the Look In field.

   b. Select the **Properties** tab.

   c. Select **Find All Files Created or Modified**.

   d. Select **During the Previous**; then enter **24** in the text field.

   e. Select **Find**.
9. Close the Find Files dialog.

10. From the terminal window command line, find all files in the home directory whose names start with “my” by entering

    find ~ -name “my*”

11. Find all files in the directory /tmp/ that were changed or created in the last 24 hours by entering

    find /tmp -ctime -1

12. Su to root (**su -**) with a password of **novell**.

13. Create the locate database by entering **updatedb**.

    Notice that the updatedb utility also searches for any files located on your cdrom or floppy drive to add to the database.

14. When the database is updated, find all files in your file system whose names contain the string “my” by entering

    locate my

15. Log out as root by entering **exit**.

16. Close the terminal window by entering **exit**.

(End of Exercise)