Directory Structure

Some of the more important and commonly used directories in the Linux directory hierarchy are listed in Table many of the directories listed in the table are also found in a UNIX file system. Table Important directories in the Linux operating system and their purpose.

/ The root directory (not to be concerned with the root account) is similar to a drive letter in Windows (C:\, D:\, etc.) except that in the Linux directory structure there is only one root directory and everything falls under it (including other file systems and partitions). The root directory is the directory that contains all other directories. When a directory structure is displayed as a tree, the root directory is at the top. Typically no files or programs are stored directly under root.

/bin This directory holds binary executable files that are essential for correct operation of the system (exactly which binaries are in this directory is often dependent upon the distribution). These binaries are usually available for use by all users. **/usr/bin** can also be used for this purpose as well.

/boot This directory includes essential system boot files including the kernel image.

/dev This directory contains the devices available to Linux. Remember that Linux treats devices like files and you can read and write to them as if they were. Everything from floppy drives to printers to your mouse is contained in this directory. Included in this directory is the notorious /dev/null, which is most useful for deleting outputs of various, functions and programs.

/etc Linux uses this directory to store system configuration files. Most files in this directory are text and can be edited with your favorite text editor. This is one of Linux's greatest advantages because there is never a hidden check box and just about all your configurations are in one place.

/etc/inittab is a text file that details what processes are started at system boot up and during regular operation. **/etc/fstab** identifies file systems and their mount points (like floppy, CD-ROM, and hard disk drives)./**etc/passwd** is where users are defined.

/home This is where every user on a Linux system will have a personal directory. If your username is "chris" then your home directory will be "/home/chris". A quick way to return to your home directory is by entering the "cd" (Change Directory) command. Your current working directory will be changed to your home directory. Usually, the permissions on user directories are set so that only root and the user the directory belongs to can access or store information inside of it. When partitioning a Linux file system this directory will typically need the most space.

/lib Shared libraries and kernel modules are stored in this directory The libraries can be dynamically linked which makes them very similar to DLL files in the Windows environment.

/lost+found This is the directory where Linux keeps files that are restored after a crash or when a partition hasn't been unmounted properly before a shutdown.

/mnt Used for mounting temporary file systems. File systems can be mounted anywhere but the /mnt directory provides a convenient place in the Linux directory structure to **mount** temporary file systems.

/opt Often used for storage of large applications packages

/proc This is a special, "virtual" directory where system processes are stored. This directory doesn't physically exist but you can often view (or read) the entries in this directory.

/root The home directory for the super user (root). Not to be confused with the root (/) directory of the Linux file system.

/sbin Utilities used for system administration (halt, ifconfig, fdisk, etc.) are stored in this directory. /usr/sbin, and /usr/local/sbin are other directories that are used for this purpose as well. **/sbin/init.d** are scripts used by /sbin/init to start the system.

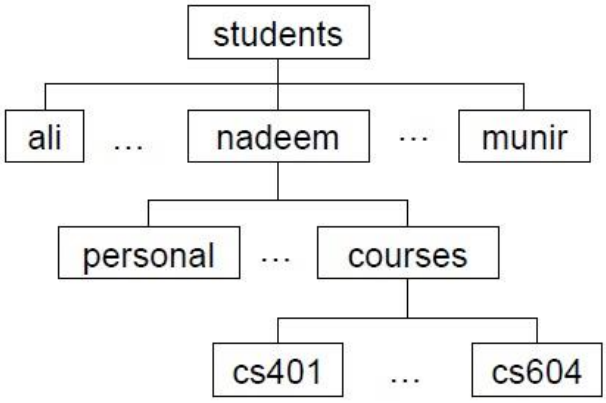
/tmp Used for storing temporary files. Similar to C:\Windows\Temp.

/usr Typically a shareable, read-only directory. Contains user applications and supporting files for those applications. **/usr/X11R6** is used by the X Window System. **/usr/bin** contains user accessible commands. **/usr/doc** holds documentation for /usr applications. **/usr/include** this directory contains header files for the C compiler. **/usr/include/g++** contains header files for the C++ compiler. **/usr/lib** libraries, binaries, and object files that aren't usually executed directly by users. **/usr/local** used for installing software locally that needs to be safe from being overwritten when system software updates occur. **/usr/man** is where the manual pages are kept. **/usr/share** is for read-only independent data files. **/usr/src** is used for storing source code of applications installed and kernel sources and headers.

/var This directory contains variable data files such as logs (**/var/log**), mail (**/var/mail**), and spools (**/var/spool**) among other things.

Browsing UNIX/Linux directory structure

We will continue that discussion and learn how to browse the UNIX/Linux directory structure. In Figure we have repeated for our reference the home directory structure for students. In the rest of this section, we discuss commands for creating directories, removing directories, and browsing the UNIX/Linux directory structure.



Displaying Directory Contents

You can display the contents (names of files and directories) of a directory with the ls command. Without an argument, it assumes your current working directory. So, if you run the ls command right after you login, it displays names of files and directories in your home directory. It does not list those files whose names start with a dot (.). Files that start with a dot are known as **hidden files** (also called dot files). You should not modify these files unless you are quite familiar with the purpose of these files and why you want to modify them. You can display all the files in a directory by using ls –a command. You can display the long listing for the contents of a directory by using the ls –l command. The following session shows sample runs of these commands.

```
$ ls
books courses LinuxKernel chatClient.c chatServer.c
$ ls -a
. .bash_history courses .login .profile
.. .bash_profile .cshrc books
chatClient.c chatServer.c LinuxKernel
$ ls –l
drwxr-xr-x 3 msarwar faculty 512 Oct 28 10:28 books
-rw-r--r-- 1 msarwar faculty 9076 Nov 4 10:14 chatClient.c
-rw-r--r-- 1 msarwar faculty 8440 Nov 4 10:16 chatServer.c
drwxr-xr-x 2 msarwar faculty 512 Feb 27 17:21 courses
drwxr-xr-x 2 msarwar faculty 512 Oct 21 14:55 LinuxKernel
$
```

The output of the ls –l command gives you the following information about a file:

- 1st character: type of a file
- Rest of letters in the 1st field: access privileges on the file
- 2nd field: number of hard links to the file
- 3rd field: owner of the file

- 4th field: Group of the owner
- 5th field: File size in bytes
- 6th and 7th fields: Date last updated
- 8th field: Time last updated
- 9th field: File name

Creating Directories

You can use the mkdir command to create a directory. In the following session, the first command creates the courses directory in your current directory. If we assume that your current directory is your home directory, this command creates the courses directory under your home directory. The second command creates the cs604 directory under the ~/courses directory (i.e., the under the courses directory under your home directory). The third command creates the programs directory under your ~/courses/cs604 directory.

$ **mkdir courses**
$ **mkdir ~/courses/cs604**
$ **mkdir ~/courses/cs604/programs**
$

You could have created all of the above directories with the **mkdir –p** ~/courses/cs604/programs command.

Removing (Deleting) Directories

You can remove (delete) an empty directory with the mkdir command. The command in the following session is used to remove the ~/courses/cs604/programs directory if it is empty.

$ **rmdir courses**
$

Changing Directory

You can jump from one directory to another (i.e., change your working directory) with the cd command. You can use the cd ~/courses/cs604/programs command to make ~/courses/ cs604/programs directory your working directory. The cd or cd $HOME command can be used to make your home directory your working directory.

Display Absolute Pathname of Your Working Directory

You can display the absolute pathname of your working directory with the pwd command, as shown below.

$ **pwd**
/home/students/nadeem/courses/cs604/programs
$

Copying, Moving, and Removing Files

We now discuss the commands to copy, move (or rename), and remove files.

Copying Files

You can use the cp command for copying files. You can use the cp file1 file2 command to copy file1 to file2. The following command can be used to copy file1 in your home directory to the ~/memos directory as file2.

$ **cp ~/file1 ~/memos/file2**
$

Moving Files

You can use the mv command for moving files. You can use the mv file1 file2 command to move file1 to file2. The following command can be used to move file1 in your home directory to the ~/memos directory as file2.

$ **mv ~/file1 ~/memos/file2**
$

Removing Files

You can use the rm command to remove files. You can use the rm file1 command to remove file1. You can use the first command the following command 28 to remove the test.c file in the ~/courses/cs604/programs directory and the second command to remove all the files with .o extension (i.e., all object files) in your working directory.

$ **rm ~/courses/cs604/programs/test.c**
$ **rm *.o**
$

Compiling and Running C Programs

You can compile your program with the gcc command. The output of the compiler command, i.e., the executable program is stored in the a.out file by default. To compile a source file titled program.c, type:

$ **gcc program.c**
$

You can run the executable program generated by this command by typing./a.out and hitting the <Enter> key, as shown in the following session.

$ **./a.out**
[... program output ...]
$

You can store the executable program in a specific file by using the –o option. For example, in the following session, the executable program is stored in the assignment file.

$ **gcc program.c –o assignment**
$

The gcc compiler does not link many libraries automatically. You can link a library explicitly by using the –l option. In the following session, we are asking the compiler to link the math library with our object file as it creates the executable file.

$ **gcc program.c –o assignment -lm**
$ **assignment**
[... program output ...]
$