



Course	Getting started with UNIX/Linux	
Module Day 1		3 Hours
	<p>At the end of this module you will be able to know about:</p> <ul style="list-style-type: none"> ▪ UNIX and its history. ▪ LINUX? History and different distributions ▪ GNU project and Free software foundation ▪ OS and its components ▪ Booting Process in Linux, ▪ Directory structure in linux and some useful commands regarding linux working. 	

Session Plan	
1	What is unix ,Brief history, Unix Versions, What is linux, Different distributions, GNU project and Free Software Foundation, OS Components - Kernel, Shell
2	system start-up, logging in and logging out, system shutdown, getting help with man pages, users and groups
3	Directory structure, paths - absolute and relative path; case sensitive Navigating the file systems - commands - cat, cd, cp, file, find, head, less, ls, mkdir, more, mv, pwd, rm, rmdir, tail, touch, where is, which

	Module Overview
---	------------------------

- ✓ What is UNIX and its history?
- ✓ Linux and its distributions
- ✓ GNU project and Free software foundation
- ✓ Operating system concepts
- ✓ System start-up – Booting process
- ✓ Users and groups
- ✓ Directory structure of linux
- ✓ Paths in linux
- ✓ Commands in linux



What is UNIX

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.



History of UNIX

Imagine computers as big as houses, even stadiums. While the sizes of those computers posed substantial problems, there was one thing that made this even worse: every computer had a different operating system.

Software was always customized to serve a specific purpose, and software for one given system didn't run on another system. Being able to work with one system didn't automatically mean that you could work with another. It was difficult, both for the users and the system administrators.

Computers were extremely expensive then, and sacrifices had to be made even after the original purchase just to get the users to understand how they worked. The total cost per unit of computing power was enormous.

Technologically the world was not quite that advanced, so they had to live with the size for another decade. In 1969, a team of developers in the Bell Labs laboratories started working on a solution for the software problem, to address these compatibility issues.

They developed a new operating system, which was

1. Simple and elegant.
2. Written in the C programming language instead of in assembly code.
3. Able to recycle code.

The Bell Labs developers named their project "UNIX." The code recycling features were very important. Until then, all commercially available computer systems were written in a

code specifically developed for one system. UNIX on the other hand needed only a small piece of that special code, which is now commonly named the kernel. This kernel is the only piece of code that needs to be adapted for every specific system and forms the base of the UNIX system. The operating system and all other functions were built around this kernel and written in a higher programming language, C.



UNIX versions

Some of the well-known UNIX flavors, are given in the below table.

Product name	Distributor / provider
AIX	IBM
BSD/OS (BSDi)	Wind River
CLIX	Intergraph Corp.
Debian GNU/Linux	Software in the Public Interest, Inc.
Tru64 UNIX (formerly Digital UNIX)	Compaq Computer Corp.
DYNIX/ptx	IBM(formerly by Sequent Computer Systems)
Esix UNIX	Esix Systems
FreeBSD	FreeBSD Group
GNU Herd	GNU Organization
HAL SPARC64/OS	HAL Computer Systems, Inc.
HP-UX	Hewlett-Packard Company
Irix	Silicon Graphics, Inc.
LynxOS	Lynx Real-Time Systems, Inc.
MacOS X Server	Apple Computer, Inc.
NetBSD	NetBSD Group
NonStop-UX	Compaq Computer Corporation
OpenBSD	OpenBSD Group
OpenLinux	Caldera Systems, Inc.
Openstep	Apple Computer, Inc.
Red Hat Linux	Red Hat Software, Inc.
Reliant UNIX	Siemens AG
SCO UNIX	The Santa Cruz Operation Inc.
Solaris	Sun Microsystems
SuSE	S.u.S.E., Inc.
UNICOS	Silicon Graphics, Inc.
UTS	UTS Global, LLC



Introduction to Linux

Linux is a UNIX-like system, but it is not UNIX. That is, although Linux borrows many ideas from UNIX and implements the UNIX API (as defined by POSIX and the Single UNIX Specification), it is not a direct descendant of the UNIX source code like other UNIX Systems.

Linux is actually the kernel, while the parts with which most people are familiar—the tools, shell, and file system—are the creations of others (usually the GNU organization).

In 1991 Linus Torvalds, a finish graduate student began work on a UNIX-like system called Linux. It began as a hobby inspired by Andy Tanenbaum's Minix, a small UNIX like system, but has grown to become a complete system in its own right. The intention is that the Linux kernel will not incorporate proprietary code but will contain nothing but freely distributable code.

Versions of Linux are now available for a wide variety of computer systems using many different types of CPUs, including PCs based on 32-bit and 64-bit Intel x86 and compatible processors; workstations and servers using Sun SPARC, IBM PowerPC, AMD Opteron, and Intel Itanium.

LINUX Distribution:

Linux is actually just a kernel. You can obtain the sources for the kernel to compile and install it on a machine and then obtain and install many other freely distributed software programs to make a complete Linux installation.

Some well-known distributions, particularly on the Intel x86 family of processors, are Red Hat Enterprise Linux and its community-developed cousin Fedora, Novell SUSE Linux and the free open SUSE variant, Ubuntu Linux, Slack ware, Gentoo, and Debian GNU/Linux. Check out the DistroWatch site at <http://distrowatch.com> for details on many more Linux distributions.



GNU Project and Free Software Foundation:

Linux owes its existence to the cooperative efforts of a large number of people. The operating system kernel itself forms only a small part of a usable development system. Commercial UNIX systems traditionally come bundled with applications that provide system services and tools. For Linux systems, these additional programs have been written by many different programmers and have been freely contributed.

The Linux community (together with others) supports the concept of free software, that is, software that is free from restrictions, subject to the GNU General Public License (the name GNU stands for the recursive *GNU's Not UNIX*).

The Free Software Foundation was set up by Richard Stallman, the author of GNU Emacs, one of the best-known text editors for UNIX and other systems. Stallman is a pioneer of the free software concept and started the GNU Project, an attempt to create an operating system and development environment that would be compatible with UNIX, but not suffer the restrictions of the proprietary UNIX name and source code.

GNU may one day turn out to be very different from UNIX in the way it handles the hardware and manages running programs, but it will still support UNIX-style applications.

The GNU Project has already provided the software community with many applications that closely mimic those found on UNIX systems. All these programs, so-called GNU software, are distributed under the terms of the GNU General Public License (GPL).

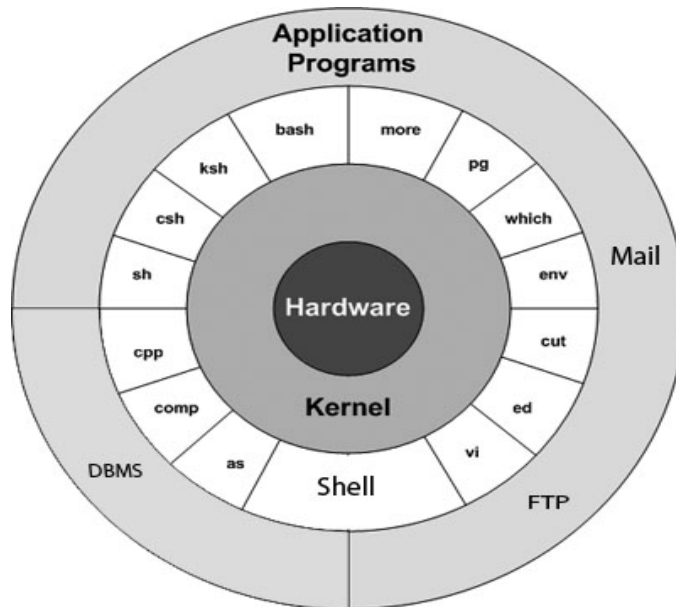
A few major examples of software from the GNU Project distributed under the GPL follow:

- GCC: The GNU Compiler Collection, containing the GNU C compiler
- G++: A C++ compiler, included as part of GCC
- GDB: A source code-level debugger
- GNU make: A version of UNIX make
- Bison: A parser generator compatible with UNIX yacc
- bash: A command shell
- GNU Emacs: A text editor and environment

You can learn more about the free software concept at <http://www.gnu.org>.



An operating system is the software interface between the user and the hardware of a system. Whether your operating system is UNIX, DOS, Windows, or OS/2, everything you do as a user or programmer interacts with the hardware in some way.



The kernel is sometimes referred to as the *supervisor*, *core* of the operating system. Typical components of a kernel are interrupt handlers to service interrupt requests, a scheduler to share processor time among multiple processes, a memory management system to manage process address spaces, and system services such as networking and inter-process communication.

The kernel typically resides in an elevated system state compared to normal user applications. This includes a protected memory space and full access to the hardware. This system state and memory space is collectively referred to as *kernel-space*.

When executing kernel code, the system is in kernel-space executing in kernel mode.

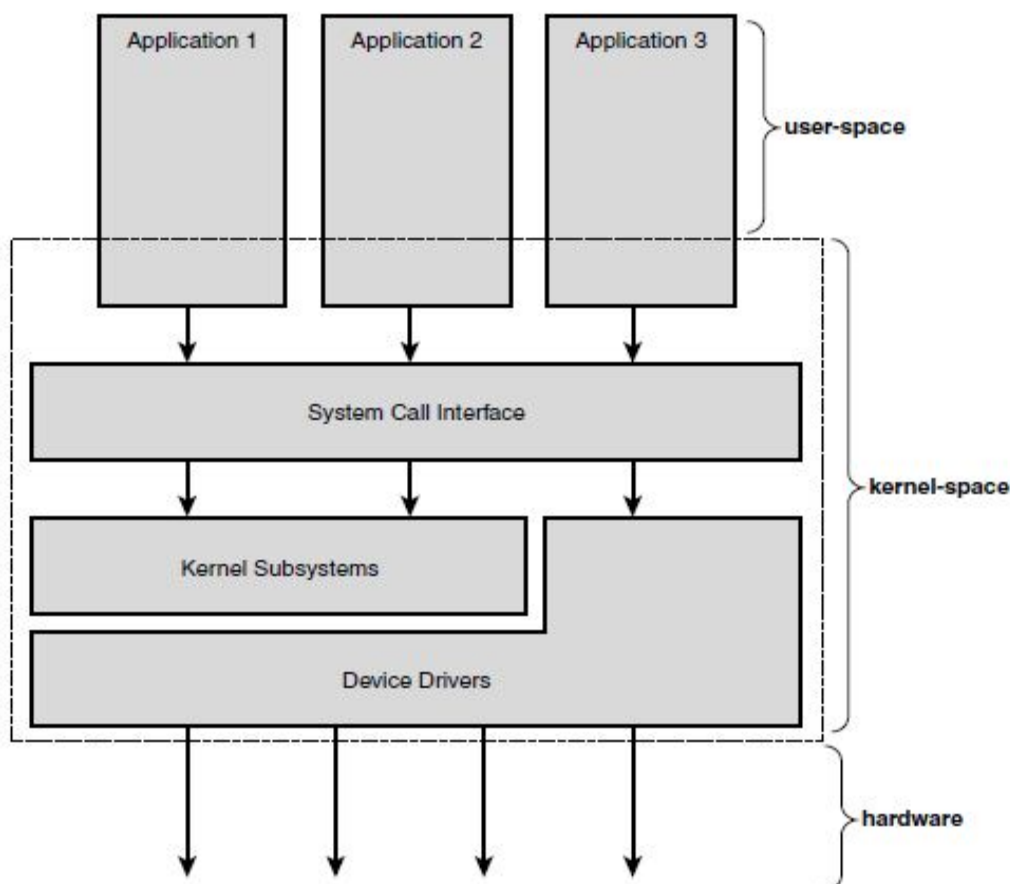
When running a regular process (User applications), the system is in user-space executing in user mode.

Applications running on the system communicate with the kernel via *system calls*.

When an application executes a system call (we say that the kernel is executing on behalf of the application). Furthermore, the application is said to be executing a system call in kernel-space, and the kernel is running in process context. This relationship—that, applications call into the kernel via the system call interface—is the fundamental manner in which applications get work done.

In Linux, we can generalize that each processor is doing exactly one of three things at any given moment:

- In user-space, executing user code in a process.
- In kernel-space, in process context, executing on behalf of a specific process.
- In kernel-space, in interrupt context, not associated with a process, handling an Interrupt.





UNIX Kernel:

The kernel is the lowest layer of the UNIX system. It provides the core capabilities of the system and allows processes (programs) to access the hardware in an orderly manner. Basically, the kernel controls processes, input/output devices, file system operations, and any other critical functions required by the operating system. It also manages memory.

When there isn't enough physical memory, the system tries to accommodate the process by moving portions of it to the hard disk. When the portion of the process that was moved to hard disk is needed again, it is returned to physical memory. This procedure, allows the system to provide multitasking capabilities, even with limited physical memory.

A kernel is built for the specific hardware on which it is operating, so a kernel built for a Sun Sparc machine can't be run on an Intel processor machine without modifications. Because the kernel deals with very low-level tasks, such as accessing the hard drive or managing multitasking, and is not user friendly, it is generally not accessed by the user.

Shell:

The shell is a command line interpreter that enables the user to interact with the operating system. A shell provides the next layer of functionality for the system; it is what you use directly to administer and run the system.



System Start-Up

What occurs from the power-off position until your operating system is fully available is called the *boot process*.

In the simplest terms, the boot process starts with Read-Only Memory's (ROM, or NVRAM, or firmware) loading of the program for actually booting (starting) the system.

This initial step (commonly called bootstrapping) identifies the devices on the system that can be booted or started from. You can boot or start from only one device at a time, but, because many different devices can be identified as bootable, one of those identified devices can be used if one bootable device has a failure.

These devices may load automatically, or you may be shown a list of devices from which you can choose.

The boot device can be a physical hard drive, the network or a removable storage media such as a CD-ROM or floppy disk.

The bootstrap phase only identifies the hardware available for booting and whether it is usable. Control is then transferred to the kernel.

After the initial bootstrapping, the boot program begins loading the UNIX kernel, which typically resides in the boot partition of the system. The kernel on most UNIX systems is called UNIX; in Linux systems, it might be called vmUNIX or vmlinuz.

The kernel's initial tasks, which vary according to hardware and UNIX version, are followed by the initialization phase, in which the system processes and scripts are started.

The init process is the first job started and is the parent of all other processes. It must be running for the system to run. The init process calls the initialization scripts and completes administrative tasks relative to the system.

Init process will create all the required processes and finally login prompt is called.



Logging In and Out of UNIX

Logging in means that you are authenticating yourself to the UNIX system as a valid user who needs to utilize resources. When you attempt to log in to a UNIX system, you are typically asked to authenticate yourself by providing a username and password pair.

You can log in by using either a graphical user interface (GUI) or the command line.

Command	Description
ssh	Logs in interactively to a shell to perform multiple functions such as running commands. This method uses encryption to scramble the session so that the username, password, and all communications with the remote system are encrypted.
telnet	Logs in interactively to a shell to perform multiple functions such as running commands. Because this method is not encrypted, the username, password and all communications with the remote system are sent in plain text and possibly viewable by others on the network.
sftp	Logs in to transfer files between two different systems. This method is encrypted.
ftp	Logs in to transfer files between two different systems. This method is not encrypted.

After you have completed the work you need to do on the system using your interactive login, you need to exit the system in a controlled and orderly manner to prevent processes or jobs from ending abruptly.

The command `exit` ends your shell session. This closes the window that you are logged in to or ends your session completely.



System Shutdown

UNIX is a multiuser, multitasking system, so there are usually many processes or programs running at all times. Because the file system needs to be synchronized, just turning the power off creates issues with the file system and affects the stability of the system.

There are always processes or tasks running on the system, even if no users are logged in, and an improper shutdown can cause numerous problems.

Command	Function
halt	Brings the system down immediately.
init 0	Powers off the system using predefined scripts to synchronize and clean up the system prior to shutdown.
init 6	Reboots the system by shutting it down completely and then bringing it completely back up.
poweroff	Shuts down the system by powering off.
reboot	Reboots the system.
shutdown	Shut downs the system.



Getting Help With MAN Page

UNIX commands have always had a multitude of arguments or options to allow different types of functionality with the same command.

UNIX's version of help files are called *man pages*. Man (manual) pages present online documentation in a standard format that is readable by any user and is set up in a consistent and logical manner. The command is used by simply typing the following syntax:

```
man man
```

```
Terminal
MAN(1) Manual pager utils MAN(1)

NAME
    man - an interface to the on-line reference manuals

SYNOPSIS
    man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L
    locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I]
    [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P
    pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justifi-
    cation] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z]
    [[section] page ...] ...
    man -k [apropos options] regex ...
    man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
    man -f [whatis options] page ...
    man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L
    locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t]
    [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
    man -w|-W [-C file] [-d] [-D] page ...
    man -c [-C file] [-d] [-D] page ...
    man [-hV]

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is
    normally the name of a program, utility or function. The manual page
    associated with each of these arguments is then found and displayed. A
    section, if provided, will direct man to look only in that section of
    the manual. The default action is to search in all of the available
    sections, following a pre-defined order and to show only the first page
    found, even if page exists in several sections.

    The table below shows the section numbers of the manual followed by the
    types of pages they contain.

    1 Executable programs or shell commands
    2 System calls (functions provided by the kernel)
    3 Library calls (functions within program libraries)

Manual page man(1) line 1 (press h for help or q to quit)
```

man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections, following a pre-defined order and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

- 1 Executable programs or shell commands
- 2 System calls (functions provided by the kernel)
- 3 Library calls (functions within program libraries)
- 4 Special files (usually found in /dev)
- 5 File formats and conventions eg /etc/passwd
- 6 Games
- 7 Miscellaneous (including macro packages and conventions), e.g. man (7), groff(7)
- 8 System administration commands (usually only for root)
- 9 Kernel routines [Non standard]



Understanding Users and Groups

There are three primary types of accounts on a UNIX system: the root user (or superuser) account, system accounts, and user accounts.

Root Account

The root account's user has complete and unfettered control of the system, to the point that he can run commands to completely destroy the system. The root user (also called root) can do absolutely anything on the system, with no restrictions on files that can be accessed, removed, and modified.

System Accounts

System accounts are those needed for the operation of system-specific components. They include, for example, the mail account (for electronic mail functions) and the sshd account (for ssh functionality). System accounts are generally provided by the operating system during installation or by a software manufacturer (including in-house developers).

User Accounts

User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories

Group Accounts

Group accounts add the capability to assemble other accounts into logical arrangements for simplification of privilege (permission) management. UNIX

permissions are placed on files and directories and are granted in three subsets: the owner of the file, also known as the user; the group assigned to the file, also known simply as group; and anyone who has a valid login to the system but does not fall into either the owner or group subsets, also known as others.

- Linux understands Users and Groups.
- A user can belong to several groups.
- A file can belong to only one user and one group at a time.
- A particular user, the superuser “root” has extra privileges.
- Only root can change the ownership of a file.
- User information are available in `/etc/passwd`.
- Group information is in `/etc/group`.



File System Concepts

A file system is a component of UNIX that enables the user to view, organize, secure, and interact with files and directories that are located on storage devices. There are different types of file systems within UNIX: disk-oriented, network-oriented, and special, or virtual.

- Disk-oriented (or local) file system—Physically accessible file systems residing on a hard drive, CD-ROM, DVD ROM, USB drive, or other device. Examples include UFS (UNIX File System), FAT (File Allocation Table, typically Windows and DOS systems), NTFS (New Technology File System, usually Windows NT, 2000, and XP systems), UDF (Universal Disk Format, typically DVD), HFS+ (Hierarchical File System, such as Mac OS X), ISO9660 (typically CD-ROM), and EXT2 (Extended Filesystem 2).
- Network-oriented (or network-based) file system—A file system accessed from a remote location. These are usually disk-oriented on the server side, and the clients access the data remotely over the network. Examples include Network File System (NFS), Samba (SMB/CIFS), AFP (Apple Filing Protocol), and WebDAV.
- Special, or virtual, file system—A file system that typically doesn't physically reside on disk, such as the TMPFS (temporary file system), PROCFS (Process File System), and LOOPBACKFS (the Loopback File System).

Everything in UNIX is considered to be a file, including physical devices such as DVD-ROMs, USB devices, floppy drives, and so forth. This use of files allows UNIX to be consistent in its treatment of resources and gives the user a consistent mechanism of interaction with the system.

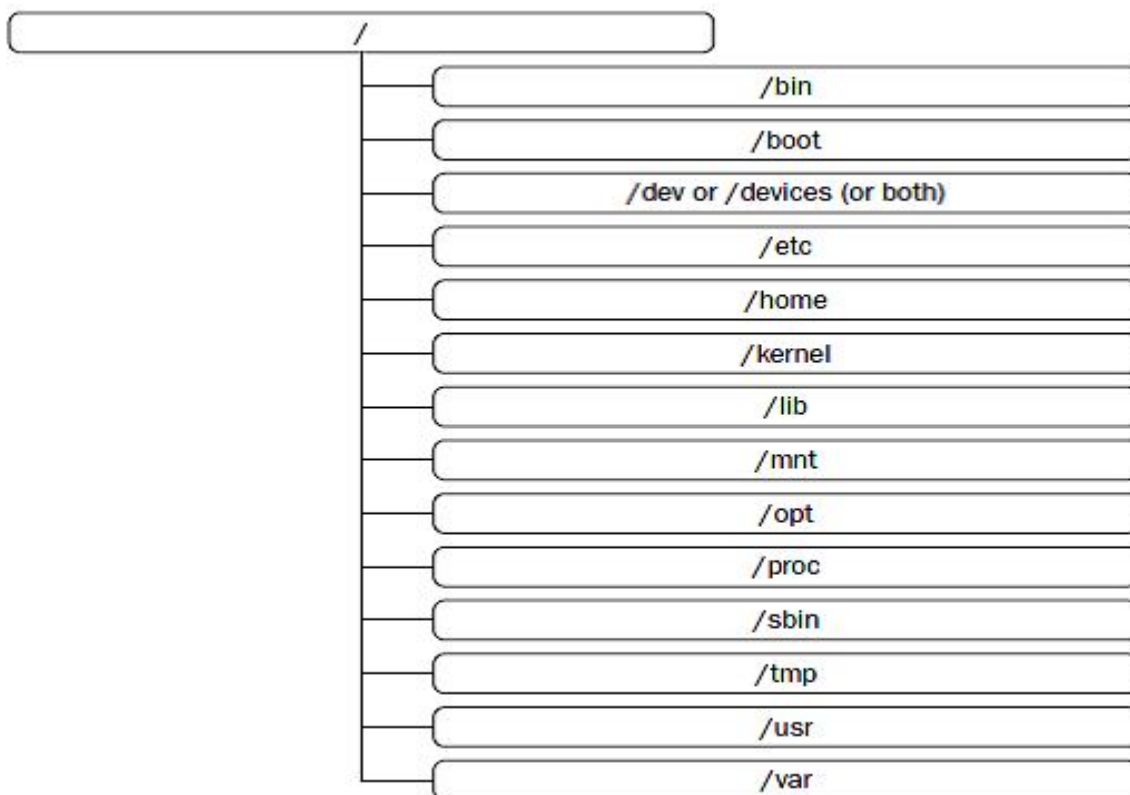
UNIX uses a hierarchical structure to organize files, providing a from-the-top approach to finding information by drilling down through successive layers in an organized fashion to locate what's needed.



Directory Structure:

In UNIX, everything starts with the root directory, often designated only by `/`.

UNIX uses a hierarchical file system structure, much like an upside-down tree, with root (`/`) at the base of the file system and all other directories spreading from there. The vast majority of UNIX systems use the directories shown in Figure.



Directory	Description
-----------	-------------

/	Root should contain only the directories needed at the top level of the file structure (or that come already installed in it). Unnecessary subdirectories under root can clutter your system, making administration more difficult and, depending on the system, filling up the space allocated for /.
bin	Usually contains binary (executable) files critical for system use, and often contains essential system programs, such as vi (for editing files), passwd (for changing passwords), and sh (the Bourne shell).
boot	Contains files for booting the system.
devices dev	Either or both of these will exist. They contain device files, often including cdrom(CD-ROM drive), eth0 (Ethernet interface), and fd0 (floppy drive). (The devices are often named differently in the different UNIX systems.)
etc	Contains system configuration files such as passwd (holds user account information and is not to be confused with /bin/passwd); hosts (contains information about host resolution); and shadow (contains encrypted passwords).
export	Often contains remote file systems (those external to the physical system), such as home directories exported from another system to save space and centralize home directories.
home	Contains the home directory for users and other accounts (specified in /etc/passwd, for example).
kernel	Contains kernel files.
lib	Contains shared library files and sometimes other kernel-related files.
mnt	Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppy diskette drive, respectively.
proc	Contains all processes marked as a file by process number or other information that is dynamic to the system.
sbin	Contains binary (executable) files, usually for system administration. Examples include fdisk (for partitioning physical disks) and ifconfig (for configuring network interfaces).
tmp	Holds temporary files used between system boots (some UNIX systems do not delete the contents of the tmp directory between boots).
usr	Used for miscellaneous purposes, or can be used by many users (such as for man pages). Can include administrative commands, shared files, library files, and others.
var	Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data. For instance, the log files (typically in /var/log) range in size from very small to very large, depending on the system configuration.

Paths and Case:

Every file has an absolute path and a relative path.

The absolute path refers to the exact location of the file in its file system, such as `/etc/passwd`.

The relative path refers to the location of a file or directory in relation (relative) to your current location. If you are in the `/etc` directory, for example, the relative path to `/etc/passwd` is `passwd` because it's in the same directory you are. This is analogous to the location of your home.

UNIX is a case-sensitive operating system. This means that the case (capitalization) of file and directory names matters.

In UNIX, you must know the case of the file or directory name because you could have three different files named `real_file`, `Real_file`, and `REAL_FILE`.

UNIX filenames are conventionally lowercase.



Navigating The File System

Command	Description
cat	Concatenate: displays a file.
cd	Change directory: moves you to the directory identified.
cp	Copy: copies one file/directory to specified location.
file	Identifies the file type (binary, text, etc).
find	Finds a file/directory.
head	Shows the beginning of a file.
less	Browses through a file from end or beginning.
ls	List: shows the contents of the directory specified.
mkdir	Make directory: creates the specified directory.
more	Browses through a file from beginning to end.
mv	Move: moves the location of or renames a file/directory.
pwd	Print working directory: shows the current directory the user is in.
rm	Remove: removes a file.
rmdir	Remove directory: removes a directory.

tail	Shows the end of a file.
touch	Creates a blank file or modifies an existing file's attributes
whereis	Shows the location of a file
which	Shows the location of a file if it is in your PATH.

File Types:

In the ls -l, every file line begins with a d, -, or l. These characters indicate the type of file that's listed. There are other file types (shown by their ls -l single-character representation in the following table), but these three are the most common.

```

utl@desktop:$ls -l
total 145044
-rwxrwxr-x 1 linux linux      20 May  9 17:31 age
-rwxrwxr-x 1 linux linux  7453 Jun  1 16:47 a.out
-rw-rw-r-- 1 linux linux    215 May 15 16:02 backgrnd.c
-rwxrwxr-x 1 linux linux    347 May  9 11:05 calc
-rw-rw-r-- 1 linux linux    229 May 14 13:12 child_delay.c
-rw-rw-r-- 1 linux linux    356 May 15 10:30 cow.c
-rw-rw-r-- 1 linux linux    896 May 18 11:05 dae.c
-rw-rw-r-- 1 linux linux    293 May 17 09:36 daemon.c
-rw-rw-r-- 1 linux linux    250 May 18 11:17 d.c
-rw-rw-r-- 1 linux linux    758 May 18 10:22 deamon.c
drwxr-xr-x 9 linux linux  4096 Jun  1 17:36 Desktop
-rw-rw-r-- 1 linux linux     54 May  9 16:11 details.out
drwxr-xr-x 2 linux linux  4096 May 29 18:34 Documents
drwxr-xr-x 3 linux linux  4096 Jun 12 09:13 Downloads
-rwxrwxr-x 1 linux linux     20 May  9 15:14 echo
-rw-r--r-- 1 linux linux  8445 May  8 19:30 examples.desktop
-rw-rw-r-- 1 linux linux  6822 May 10 12:08 excl.out
-rw-rw-r-- 1 linux linux  7599 May 10 12:08 exc.out
-rw-rw-r-- 1 linux linux    121 May 17 15:33 exec.c
-rw-rw-r-- 1 linux linux    184 May 14 12:22 exec_ins.c
-rwxrwxr-x 1 linux linux    137 May  9 09:36 fibonacci
-rw-rw-r-- 1 linux linux    166 May 18 12:11 file.c
-rw-rw-r-- 1 linux linux    215 May 15 16:03 foregrnd.c
-rwxrwxr-x 1 linux linux    149 May 10 11:16 ifelse
-rwxrwxr-x 1 linux linux    219 May  9 16:11 input

```

File Type	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link (links are discussed in the following section)
b	Block special file (block input/output device file used for transferring data from or to a device such as a physical hard drive)
c	Character special file (raw input/output device file used for transferring data from or to a device such as a physical hard drive)
d	Directory file (file that contains a listing of other files and/or directories contained within the directory)
l	Symbolic link file (discussed in the following section)
p	Named pipe (a mechanism for interprocess communications)
s	Socket (used for interprocess communication)

Key Learning: