UNIX / LINUX - FILE SYSTEM BASICS

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired.

Your hard drive can have various partitions which usually contain only one file system, such as one file system housing the **/file system** or another containing the **/home file system**.

One file system per partition allows for the logical maintenance and management of differing file systems.

Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

Directory Structure

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.

A Unix filesystem is a collection of files and directories that has the following properties –

- It has a root directory (/) that contains other files and directories.
- Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an **inode**.
- By convention, the root directory has an **inode** number of **2** and the **lost+found** directory has an **inode** number of **3**. Inode numbers **0** and **1** are not used. File inode numbers can be seen by specifying the **-i option** to **ls command**.
- It is self-contained. There are no dependencies between one filesystem and another.

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix –

Sr.No.	Directory & Description
1	**/** This is the root directory which should contain only the directories needed at the top level of the file structure
2	**/bin** This is where the executable files are located. These files are available to all users
3	**/dev** These are device drivers
4	**/etc**

	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages
5	**/lib** Contains shared library files and sometimes other kernel-related files
6	**/boot** Contains files for booting the system
7	**/home** Contains the home directory for users and other accounts
8	**/mnt** Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively
9	**/proc** Contains all processes marked as a file by **process number** or other information that is dynamic to the system
10	**/tmp** Holds temporary files used between system boots
11	**/usr** Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others
12	**/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
13	**/sbin** Contains binary (executable) files, usually for system administration. For example, ***fdisk*** and ***ifconfig*** utlities
14	

	/kernel Contains kernel files

Navigating the File System

Now that you understand the basics of the file system, you can begin navigating to the files you need. The following commands are used to navigate the system –

Sr.No.	Command & Description
1	**cat filename** Displays a filename
2	**cd dirname** Moves you to the identified directory
3	**cp file1 file2** Copies one file/directory to the specified location
4	**file filename** Identifies the file type (binary, text, etc)
5	**find filename dir** Finds a file/directory
6	**head filename** Shows the beginning of a file
7	**less filename** Browses through a file from the end or the beginning
8	**ls dirname** Shows the contents of the directory specified
9	**mkdir dirname**

	Creates the specified directory
10	**more filename** Browses through a file from the beginning to the end
11	**mv file1 file2** Moves the location of, or renames a file/directory
12	**pwd** Shows the current directory the user is in
13	**rm filename** Removes a file
14	**rmdir dirname** Removes a directory
15	**tail filename** Shows the end of a file
16	**touch filename** Creates a blank file or modifies an existing file or its attributes
17	**whereis filename** Shows the location of a file
18	**which filename** Shows the location of a file if it is in your PATH

You can use Manpage Help to check complete syntax for each command mentioned here.

The df Command

The first way to manage your partition space is with the **df (disk free)** command. The command **df -k (disk free)** displays the **disk space usage in kilobytes**, as shown below –

```
$df -k
Filesystem      1K-blocks      Used   Available Use% Mounted on
/dev/vzfs        10485760   7836644     2649116  75% /
/devices                0         0           0   0% /devices
$
```

Some of the directories, such as **/devices**, shows 0 in the kbytes, used, and avail columns as well as 0% for capacity. These are special (or virtual) file systems, and although they reside on the disk under /, by themselves they do not consume disk space.

The **df -k** output is generally the same on all Unix systems. Here's what it usually includes –

Sr.No.	Column & Description
1	**Filesystem** The physical file system name
2	**kbytes** Total kilobytes of space available on the storage medium
3	**used** Total kilobytes of space used (by files)
4	**avail** Total kilobytes available for use
5	**capacity** Percentage of total space used by files
6	**Mounted on** What the file system is mounted on

You can use the **-h (human readable) option** to display the output in a format that shows the size in easier-to-understand notation.

The du Command

The **du (disk usage) command** enables you to specify directories to show disk space usage on a particular directory.

This command is helpful if you want to determine how much space a particular directory is taking. The following command displays number of blocks consumed by each directory. A single block may take either 512 Bytes or 1 Kilo Byte depending on your system.

```
$du /etc
10      /etc/cron.d
126     /etc/default
6       /etc/dfs
...
$
```

The **-h** option makes the output easier to comprehend –

```
$du -h /etc
5k     /etc/cron.d
63k    /etc/default
3k     /etc/dfs
...
$
```

Mounting the File System

A file system must be mounted in order to be usable by the system. To see what is currently mounted (available for use) on your system, use the following command –

```
$ mount
/dev/vzfs on / type reiserfs (rw,usrquota,grpquota)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
$
```

The **/mnt** directory, by the Unix convention, is where temporary mounts (such as CDROM drives, remote network drives, and floppy drives) are located. If you need to mount a file system, you can use the mount command with the following syntax –

```
mount -t file_system_type device_to_mount directory_to_mount_to
```

For example, if you want to mount a **CD-ROM** to the directory **/mnt/cdrom**, you can type –

```
$ mount -t iso9660 /dev/cdrom /mnt/cdrom
```

This assumes that your CD-ROM device is called **/dev/cdrom** and that you want to mount it to **/mnt/cdrom**. Refer to the mount man page for more specific information or type mount **-h** at the command line for help information.

After mounting, you can use the cd command to navigate the newly available file system through the mount point you just made.

Unmounting the File System

To unmount (remove) the file system from your system, use the **umount** command by identifying the mount point or device.

For example, **to unmount cdrom**, use the following command –

```
$ umount /dev/cdrom
```

The **mount command** enables you to access your file systems, but on most modern Unix systems, the **automount function** makes this process invisible to the user and requires no intervention.

User and Group Quotas

The user and group quotas provide the mechanisms by which the amount of space used by a single user or all users within a specific group can be limited to a value defined by the administrator.

Quotas operate around two limits that allow the user to take some action if the amount of space or number of disk blocks

start to exceed the administrator defined limits –

- **Soft Limit –** If the user exceeds the limit defined, there is a grace period that allows the user to free up some space.
- **Hard Limit –** When the hard limit is reached, regardless of the grace period, no further files or blocks can be allocated.

There are a number of commands to administer quotas –

Sr.No.	Command & Description
1	**quota** Displays disk usage and limits for a user of group
2	**edquota** This is a quota editor. Users or Groups quota can be edited using this command
3	**quotacheck** Scans a filesystem for disk usage, creates, checks and repairs quota files
4	**setquota** This is a command line quota editor
5	**quotaon** This announces to the system that disk quotas should be enabled on one or more filesystems
6	**quotaoff** This announces to the system that disk quotas should be disabled for one or more filesystems
7	**repquota** This prints a summary of the disc usage and quotas for the specified file systems

You can use Manpage Help to check complete syntax for each command mentioned here.