Linux Structure

Linux is a layered operating system. The innermost layer is the hardware that provides the services for the OS. The operating system, referred to in Linux as the kernel, interacts directly with the hardware and provides the services to the user programs. These user programs don't need to know anything about the hardware. They just need to know how to interact with the kernel and it's up to the kernel to provide the desired service. One of the big appeals of Linux to programmers has been that most well written user programs are independent of the underlying hardware, making them readily portable to new systems.

User programs interact with the kernel through a set of standard system calls. These system calls request services to be provided by the kernel. Such services would include accessing a file: open close, read, write, link, or execute a file; starting or updating accounting records; changing ownership of a file or directory; changing to a new directory; creating, suspending, or killing a process; enabling access to hardware devices; and setting limits on system resources.

Linux is a multi-user, multi-tasking operating system. You can have many users logged into a system simultaneously, each running many programs. It's the kernel's job to keep each process and user separate and to regulate access to system hardware, including cpu, memory, disk and other I/O devices.

Linux Boot Sequence

1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
        initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a 'ps -ef | grep init' and check the pid.
- initrd stands for Initial RAM Disk.

- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
    - 0 – halt
    - 1 – Single user mode
    - 2 – Multiuser, without NFS
    - 3 – Full multiuser mode
    - 4 – unused
    - 5 – X11
    - 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute 'grep initdefault /etc/inittab' on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail …. OK". Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
    - Run level 0 – /etc/rc.d/rc0.d/
    - Run level 1 – /etc/rc.d/rc1.d/
    - Run level 2 – /etc/rc.d/rc2.d/
    - Run level 3 – /etc/rc.d/rc3.d/
    - Run level 4 – /etc/rc.d/rc4.d/
    - Run level 5 – /etc/rc.d/rc5.d/
    - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

Linux vs. Windows

Linux and Windows. Each has its own set of unique features, advantages and disadvantages. While it is difficult to say which one is the better choice, it is not as difficult to answer which is the better choice given your needs.

Note: The operating system that you use on your desktop computer (the vast majority of people use some flavor of Windows) has absolutely nothing to do with the one that your host needs to serve your web site. Most personal sites are created with MS FrontPage and even although that is a Microsoft product, it can be hosted perfectly on a LINUX web server with FrontPage Extensions installed.

Stability:
LINUX systems (we actually use Linux but for comparison purposes they are identical) are hands-down the winner in this category. There are many factors here but to name just a couple big ones: in our experience LINUX handles high server loads better than Windows and LINUX machines seldom require reboots while Windows is constantly needing them. Servers running on LINUX enjoy extremely high up-time and high availability/reliability.

Performance:
While there is some debate about which operating system performs better, in our experience both perform comparably in low-stress conditions however LINUX servers under high load (which is what is important) are superior to Windows.

Scalability:
Web sites usually change over time. They start off small and grow as the needs of the person or organization running them grow. While both platforms can often adapt to your growing needs, Windows hosting is more easily made compatible with LINUX-based programming features like PHP and MySQL. LINUX-based web software is not always 100% compatible with Microsoft technologies like .NET and VB development. Therefore if you wish to use these, you should choose Windows web hosting.

Compatibility:
Web sites designed and programmed to be served under a LINUX-based web server can easily be hosted on a Windows server, whereas the reverse is not always true. This makes programming for LINUX the better choice.

Price:
Servers hosting your web site require operating systems and licenses just like everyone else. Windows 2003 and other related applications like SQL Server each cost a significant amount of money; on the other hand, Linux is a free operating system to download, install and operate. Windows hosting results in being a more expensive platform.

Conclusion:

To sum it up, LINUX-based hosting is more stable, performs faster and more compatible than Windows-based hosting. You only need Windows hosting if you are going to developing in .NET or Visual Basic, or some other application that limits your choices

Logging On To System

- Before you can begin to use the system you will need to have a valid username and a password. Assignment of usernames and initial passwords is typically handled by the System Administrator
- Your username, also called a userid, should be unique and should not change. Initial passwords can be anything and should be changed after your first login.

To login to your account

- Type your username at the login prompt, initial of your first name followed by last name (e.g iafzal). LINUX is case sensitive - if your username is kellyk do not type KellyK . Press the RETURN or ENTER key after typing your username.
- When the password prompt appears, type in your password. Your password is never displayed on the screen as a security measure. It also is case sensitive. Press the RETURN or ENTER key after entering your password.
- What happens after you successfully login depends upon your system, many LINUX systems will display a login banner or "message of the day". Make a habit of reading this since it may contain important information about the system.
- Other LINUX systems will automatically configure your environment and open one or more windows for you to do work in.
- You should see a prompt - usually a percent sign (%) or dollar sign ($). This is called the "shell prompt" (the shell is discussed in detail later). It indicates that the system is ready to accept commands from you.

If your login attempt was unsuccessful, there are several possible reasons:

- You made a typing error while entering your username or password
- The CAPS LOCK key is on and everything is being sent to the system in uppercase letters.
- You have an expired or invalid username or password, or the system security has changed
- There are system problems

Example of user login

```
login: kellyk
kellyk's Password:
**********************************************************
* Welcome to the Linux Systems Training Class
**********************************************************
*
```

```
* Hello! (Greetings)
*
* System maintenance is scheduled today from 2:00
* until 4:00 pm EST
*
*                      (Thank you very much)
*
************************************************************
```

Your Home Directory

- Each user has a unique "home" directory. Your home directory is that part of the file system reserved for your files.
- After login, you are "put" into your home directory automatically. This is where you start your work.
- You are in control of your home directory and the files which reside there. You are also in control of the file access permissions (discussed later) to the files in your home directory. Generally, you alone should be able to create/delete/modify files in your home directory. Others may have permission to read or execute your files as you determine.
- In most LINUX systems, you can "move around" or navigate to other parts of the file system outside of your home directory. This depends upon how the file permissions have been set by others and/or the System Administrator

Linux File System

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 contains 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 Linux is considered to be a file, including physical devices such as DVD-ROMs, USB devices, floppy drives, and so forth.

Directory Structure:
Linux 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 LINUX 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 any other.

File System:

What are filesystems?
A filesystem is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the filesystem. Thus, one might say ``I have two filesystems'' meaning one has two partitions on which one stores files, or that one is using the ``extended filesystem'', meaning the type of the filesystem.

The difference between a disk or partition and the filesystem it contains is important. A few programs (including, reasonably enough, programs that create filesystems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong types).

Before a partition or disk can be used as a filesystem, it needs to be initialized, and the bookkeeping data structures need to be written to the disk. This process is called making a filesystem.

Most LINUX filesystem types have a similar general structure, although the exact details vary quite a bit. The central concepts are superblock, inode , data block, directory block , and indirection block. The superblock contains information about the filesystem as a whole, such as its size (the exact information here depends on the filesystem). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first.

LINUX filesystems usually allow one to create a hole in a file (this is done with the lseek() system call; check the manual page), which means that the filesystem just pretends that at a particular place in the file there is just zero bytes, but no actual disk sectors are reserved for that place in the file (this means that the file will use a bit less disk space). This happens especially often for small binaries, Linux shared libraries, some databases, and a few other special cases. (Holes are implemented by storing a special value as the address of the data block in the indirect block or inode. This special address means that no data block is allocated for that part of the file, ergo, there is a hole in the file.)

Comparing Filesystem Features

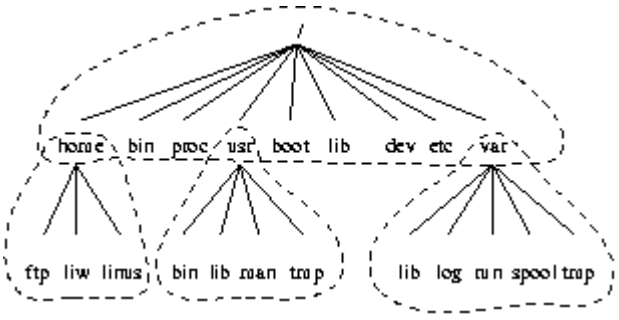
FS Name	Year Introduced	Original OS	Max File Size	Max FS Size	Journaling
FAT16	1983	MSDOS V2	4GB	16MB to 8GB	N
FAT32	1997	Windows 95	4GB	8GB to 2TB	N

FS Name	Year Introduced	Original OS	Max File Size	Max FS Size	Journaling
HPFS	1988	OS/2	4GB	2TB	N
NTFS	1993	Windows NT	16EB	16EB	Y
HFS+	1998	Mac OS	8EB	?	N
UFS2	2002	FreeBSD	512GB to 32PB	1YB	N
ext2	1993	Linux	16GB to 2TB4	2TB to 32TB	N
ext3	1999	Linux	16GB to 2TB4	2TB to 32TB	Y
ReiserFS3	2001	Linux	8TB8	16TB	Y
ReiserFS4	2005	Linux	?	?	Y
XFS	1994	IRIX	9EB	9EB	Y
JFS	?	AIX	8EB	512TB to 4PB	Y
VxFS	1991	SVR4.0	16EB	?	Y
ZFS	2004	Solaris 10	1YB	16EB	N

This topic is loosely based on the *Filesystems Hierarchy Standard* (FHS), which attempts to set a standard for how the directory tree in a Linux system should be organized. Such a standard has the advantage that it will be easier to write or port software for Linux, and to administer Linux machines, since everything should be in standardized places. There is no authority behind the standard that forces anyone to comply with it, but it has gained the support of many Linux distributions. It is not a good idea to break with the FHS without very compelling reasons. The FHS attempts to follow Linux tradition and current trends, making Linux systems familiar to those with experience with other Linux systems, and vice versa.

The full directory tree is intended to be breakable into smaller parts, each capable of being on its own disk or partition, to accommodate to disk size limits and to ease backup and other system administration tasks. The major parts are the root (/), /usr , /var , and /home filesystems (*see the following figure*). Each part has a different purpose. The directory tree has been designed so that it works well in a network of Linux machines which may share some parts of the filesystems over a read-only device (e.g., a CD-ROM), or over the network with NFS.

Parts of a Linux directory tree. Dashed lines indicate partition limits



The roles of the different parts of the directory tree are described below

- The root filesystem is specific for each machine (it is generally stored on a local disk, although it could be a ramdisk or network drive as well) and contains the files that are necessary for booting the system up, and to bring it up to such a state that the other filesystems may be mounted. The contents of the root filesystem will therefore be sufficient for the single user state. It will also contain tools for fixing a broken system, and for recovering lost files from backups.

- The /usr filesystem contains all commands, libraries, manual pages, and other unchanging files needed during normal operation. No files in /usr should be specific for any given machine, nor should they be modified during normal use. This allows the files to be shared over the network, which can be cost-effective since it saves disk space (there can easily be hundreds of megabytes, increasingly multiple gigabytes in /usr). It can make administration easier (only the master /usr needs to be changed when updating an application, not each machine separately) to have /usr network mounted. Even if the filesystem is on a local disk, it could be mounted read-only, to lessen the chance of filesystem corruption during a crash.

- The /var filesystem contains files that change, such as spool directories (for mail, news, printers, etc), log files, formatted manual pages, and temporary files. Traditionally everything in /var has been somewhere below /usr , but that made it impossible to mount /usr read-only.

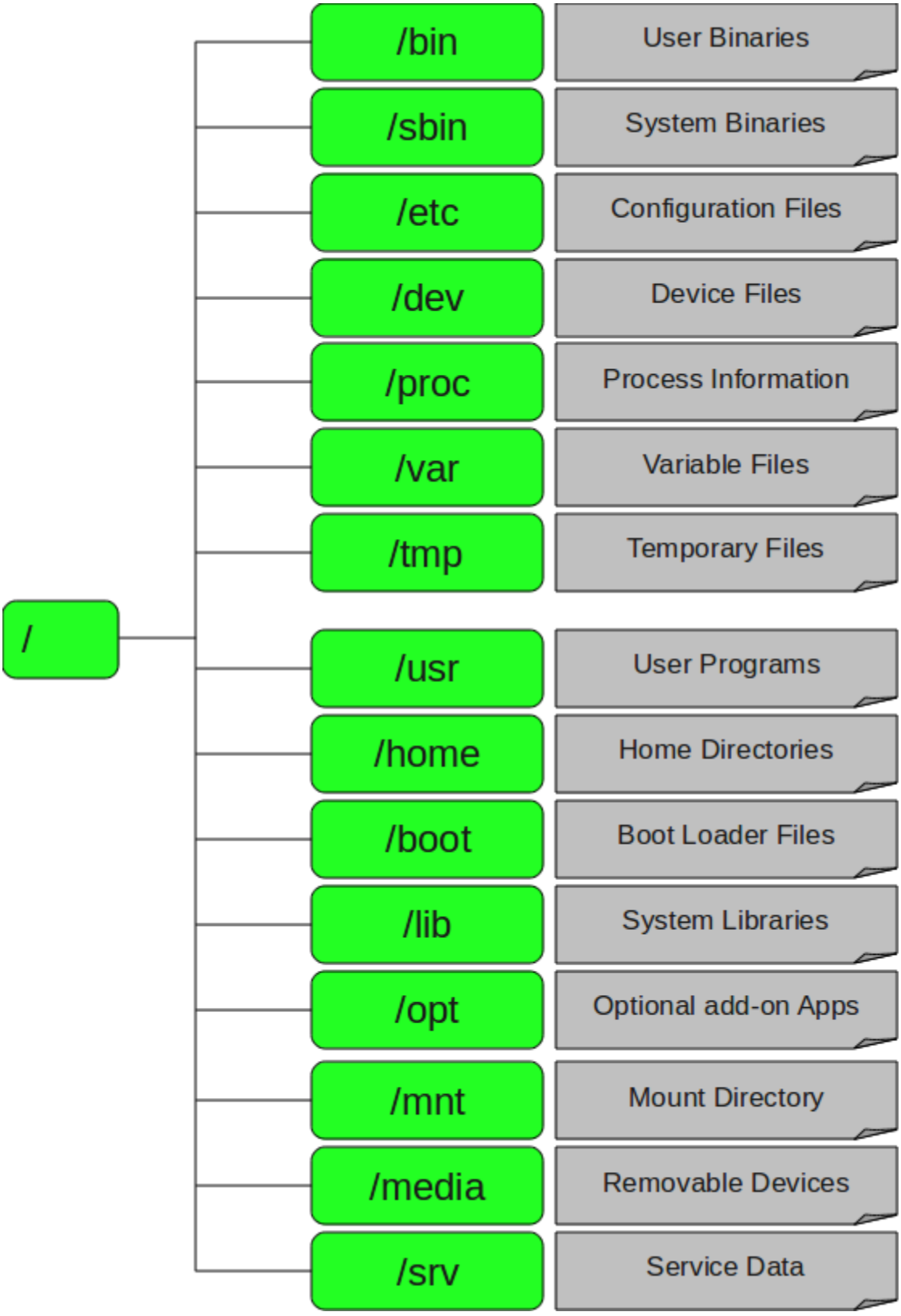
- The /home filesystem contains the users' home directories, i.e., all the real data on the system. Separating home directories to their own directory tree or filesystem makes backups easier; the other parts often do not have to be backed up, or at least not as often as they seldom change. A big /home might have to be broken across several filesystems, which requires adding an extra naming level below /home, for example /home/students and /home/staff.

  Although the different parts have been called filesystems above, there is no requirement that they actually be on separate filesystems. They could easily be kept in a single one if the system is a small single-user system and the user wants to keep things simple. The directory tree might also be divided into filesystems differently, depending on how large the disks are, and how space is allocated for various purposes. The important part, though, is that all the standard *names* work; even if, say, /var and /usr are actually on the same partition, the names /usr/lib/libc.a and /var/log/messages must work, for example by moving files below /var into /usr/var, and making /var a symlink to /usr/var.

  The Linux filesystem structure groups files according to purpose, i.e., all commands are in one place, all data files in another, documentation in a third, and so on. An alternative would be to group files according to the program they belong to, i.e., all Emacs files would be in one directory, all TeX in another, and so on. The problem with the latter approach is that it makes it difficult to share files (the program directory often contains both static and sharable and changing and non-sharable files), and sometimes to even find the files (e.g., manual pages in a huge number of places, and making the manual page programs find all of them is a maintenance nightmare).

  The root filesystem should generally be small, since it contains very critical files and a small, infrequently modified filesystem has a better chance of not getting corrupted. A corrupted root filesystem will generally mean that the system becomes unbootable except with special measures (e.g., from a floppy), so you don't want to risk it.

The root directory generally doesn't contain any files, except perhaps on older systems where the standard boot image for the system, usually called /vmlinuz was kept there. (Most distributions have moved those files the the /boot directory.


/
/bin
User Binaries
/sbin
System Binaries
/etc
Configuration Files
/dev
Device Files
/proc
Process Information
/var
Variable Files
/tmp
Temporary Files
/usr
User Programs
/home
Home Directories
/boot
Boot Loader Files
/lib
System Libraries
/opt
Optional add-on Apps
/mnt
Mount Directory
/media
Removable Devices
/srv
Service Data

1. / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

3. /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

- var stands for variable files.

- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data

File Names

- LINUX permits file names to use most characters, but avoid spaces, tabs and characters that have a special meaning to the shell, such as:

  ```
  & ; ( ) | ? \ ' " ` [ ] { } < > $ - ! /
  ```

- Case Sensitivity: uppercase and lowercase are not the same! These are three different files:

  ```
  NOVEMBER       November     november
  ```

- Length: can be up to 256 characters

- Extensions: may be used to identify types of files

  ```
  libc.a       - archive, library file
  program.c    - C language source file
  alpha2.f     - Fortran source file
  xwd2ps.o     - Object/executable code
  mygames.Z    - Compressed file
  ```

- Hidden Files: have names that begin with a dot (.) For example:

  ```
  .cshrc      .login      .mailrc     .mwmrc
  ```

- Uniqueness: as children in a family, no two files with the same parent directory can have the same name. Files located in separate directories can have identical names.

- Reserved Filenames:

  ```
  /   - the root directory (slash)
  .   - current directory (period)
  ..  - parent directory (double period)
  ~   - your home directory (tilde)
  ```

Passwords Standards

When your account is issued, you will be given an initial password. It is important for system and personal security that the password for your account be changed to something of your choosing. The command for changing a password is "passwd". You will be asked both for your old password and to type your new selected password twice. If you mistype your old password or do not type your new password the same way twice, the system will indicate that the password has not been changed. Some system administrators have installed programs that check for appropriateness of password (is it cryptic enough for reasonable system security). A password change may be rejected by this program. When choosing a password, it is important that it be something that could not be guessed -- either by somebody unknown to you trying to break in, or by an acquaintance who knows you. Suggestions for choosing and using a password follow:

Don't
use a word (or words) in any language
use a proper name
use information that can be found in your wallet
use information commonly known about you (car license, pet name, etc)
use control characters. Some systems can't handle them
write your password anywhere
ever give your password to *anybody*

Do
use a mixture of character types (alphabetic, numeric, special)
use a mixture of upper case and lower case
use at least 6 characters
choose a password you can remember
change your password often
make sure nobody is looking over your shoulder when you are entering your password

Change Password in LINUX

How do I change the password in LINUX?

To modify a user's password or your own password in LINUX use the passwd command. Open the terminal and then type the passwd command entering the new password, the characters entered do not display on screen, in order to avoid the password being seen by a passer-by. The passwd command prompts for the new password twice in order to detect any typing errors. The encrypted password is stored in /etc/shadow file.

Change Any Users Password
Login as the root user and type the command:

```
# passwd userName
# passwd vivek
# passwd tom
```

Sample outputs:

```
Enter new LINUX password:
Retype new LINUX password:
passwd: password updated successfully
```

Change Your Own Password
Simply type the passwd command:

```
$ passwd
```

Sample outputs:

```
(current) LINUX password:
Enter new LINUX password:
Retype new LINUX password:
passwd: password updated successfully
```

Difference between locate and find command in Linux

Two popular commands for locating files on Linux are find and locate. Depending on the size of your file system and the depth of your search, the find command can sometime take a long time to scan all of the data. For example, if you search your entire filesystem for the files named data.txt:

```
# find / -name data.txt
```

More likely than not, this will take on the order of minutes, if not longer to return. A quicker method is to use the locate command:

```
# locate data.txt
```

However, this efficiency comes at a cost, the data reported in the output of locate isn't as fresh as the data reported by the find command. By default, the system will run updatedb which takes a snapshot of the system files once a day, locate uses this snapshot to quickly report what files are where. However, recent file additions or removals (within 24 hours) are not recorded in the snapshot and are unknown to locate.

The find command has a number of options and is very configurable. There are many ways to reduce the depth and breadth of your search and make it more efficient.

locate uses a previously built database, If database is not updated then locate command will not show the output. to sync the database it is must to execute updatedb command.

```
# updatedb
```

How to Use Wildcards

A wildcard is a character that can be used as a substitute for any of a class of characters in a search, thereby greatly increasing the flexibility and efficiency of searches.

Wildcards are commonly used in shell commands in Linux and other Unix-like operating systems. A shell is a program that provides a text-only user interface and whose main function is to execute commands typed in by users and display their results.

Wildcards are also used in regular expressions and programming languages. Regular expressions are a pattern matching system that uses strings (i.e., sequences of characters) constructed according to pre-defined syntax rules to find desired strings in text.

The term wildcard or wild card was originally used in card games to describe a card that can be assigned any value that its holder desires. However, its usage has spread so that it is now used to describe an unknown or unpredictable factor in a variety of fields.

Star Wildcard

Three types of wildcards are used with Linux commands. The most frequently employed and usually the most useful is the star wildcard, which is the same as an asterisk (*). The star wildcard has the broadest meaning of any of the wildcards, as it can represent zero characters, all single characters or any string.

As an example, the file command provides information about any filesystem object (i.e., file, directory or link) that is provided to it as an argument (i.e., input). Because the star wildcard represents every string, it can be used as the argument for file to return information about every object in the specified directory. Thus, the following would display information about every object in the current directory (i.e., the directory in which the user is currently working):

```
file *
```

If there are no matches, an error message is returned, such as *: can't stat `*' (No such file or directory).. In the case of this example, the only way that there would be no matches is if the directory were empty.

Wildcards can be combined with other characters to represent parts of strings. For example, to represent any filesystem object that has a .jpg filename extension, *.jpg would be used. Likewise, a* would represent all objects that begin with a lower case (i.e., small) letter a.

As another example, the following would tell the ls command (which is used to list files) to provide the names of all files in the current directory that have an .html or a .txt extension:

```
ls *.html *.txt
```

Likewise, the following would tell the rm command (which is used to remove files and directories) to delete all files in the current directory that have the string xxx in their name:

```
rm *xxx*
```

Question Mark Wildcard

The question mark (?) is used as a wildcard character in shell commands to represent exactly one character, which can be any single character. Thus, two question marks in succession would represent any two characters in succession, and three question marks in succession would represent any string consisting of three characters.

Thus, for example, the following would return data on all objects in the current directory whose names, inclusive of any extensions, are exactly three characters in length:

```
file ???
```

And the following would provide data on all objects whose names are one, two or three characters in length:

```
file ? ?? ???
```

As is the case with the star wildcard, the question mark wildcard can be used in combination with other characters. For example, the following would provide information about all objects in the current directory that begin with the letter a and are five characters in length:

```
file a????
```

The question mark wildcard can also be used in combination with other wildcards when separated by some other character. For example, the following would return a list of all files in the current directory that have a three-character filename extension:

```
ls *.???
```

Square Brackets Wildcard

The third type of wildcard in shell commands is a pair of square brackets, which can represent any of the characters enclosed in the brackets. Thus, for example, the following would provide information about all objects in the current directory that have an x, y and/or z in them:

```
file *[xyz]*
```

And the following would list all files that had an extension that begins with x, y or z:

```
ls *.[xyz]*
```

The same results can be achieved by merely using the star and question mark wildcards. However, it is clearly more efficient to use the bracket wildcard.

When a hyphen is used between two characters in the square brackets wildcard, it indicates a range inclusive of those two characters. For example, the following would provide information about all of the objects in the current directory that begin with any letter from a through f:

```
file [a-f]*
```

And the following would provide information about every object in the current directory whose name includes at least one numeral:

```
file *[0-9]*
```

The use of the square brackets to indicate a range can be combined with its use to indicate a list. Thus, for example, the following would provide information about all filesystem objects whose names begin with any letter from a through c or begin with s or t:

```
file [a-cst]*
```

Likewise, multiple sets of ranges can be specified. Thus, for instance, the following would return information about all objects whose names begin with the first three or the final three lower case letters of the alphabet:

```
file [a-cx-z]*
```

Sometimes it can be useful to have a succession of square bracket wildcards. For example, the following would display all filenames in the current directory that consist of jones followed by a three-digit number:

```
ls jones[0-9][0-9][0-9]
```

Other Wild Cards

\ (backslash) = is used as an "escape" character, i.e. to protect a subsequent special character. Thus, "\ \" searches for a backslash. Note you may need to use quotation marks and backslash(es).

^ (caret) = means "the beginning of the line". So "^a" means find a line starting with an "a".

$ (dollar sign) = means "the end of the line". So "a$" means find a line ending with an "a".

For example, this command searches the file myfile for lines starting with an "s" and ending with an "n", and prints them to the standard output (screen):

```
cat myfile | grep '^s.*n$'
```

Soft Link and Hard Links

Example:
Create two files:
$ touch blah1
$ touch blah2

Enter some data into them:

$ echo "Cat" > blah1
$ echo "Dog" > blah2

And as expected:

$cat blah1; cat blah2
Cat
Dog

Let's create hard and soft links:

$ ln blah1 blah1-hard
$ ln -s blah2 blah2-soft

Let's see what just happened:

$ ls -l

blah1
blah1-hard
blah2
blah2-soft -> blah2

Changing the name of blah1 does not matter:

$ mv blah1 blah1-new
$ cat blah1-hard
Cat

blah1-hard points to the inode, the contents, of the file - that wasn't changed.

$ mv blah2 blah2-new
$ ls blah2-soft
blah2-soft
$ cat blah2-soft

cat: blah2-soft: No such file or directory

The contents of the file could not be found because the soft link points to the name, that was changed, and not to the contents.
Similarly, If blah1 is deleted, blah1-hard still holds the contents; if blah2 is deleted, blah2-soft is just a link to a non-existing file.

List folders and files in a directory

Written By: Alexandros Mavridis

The command:	Information Commands:
ls - list directory contents	ls --version ls --help info ls man ls

Contents

Options Used In This Document

-r, --reverse
reverse order while sorting

-l use a long listing format

-t sort by modification time, newest first

-i, --inode
print the index number of each file

-a, --all
do not ignore entries starting with .

-d, --directory
list directories themselves, not their contents

-p, --indicator-style=slash
append / indicator to directories

--group-directories-first
group directories before files

All the commands in the current document can be | to wc -l command for printing number of folders or files, instead of folders and files themselves. For example:

ls -d */ | wc -l

A. Listing Folders

Non hidden folders

Command	Output
ls -d */	Prints all non hidden folders in the current working directory in alphabetical order.
ls -dr */	Prints all non hidden folders in the current working directory in reverse alphabetical order.

ls -dl */	Prints in detail all non hidden folders in the current working directory in alphabetical order.
ls -l | grep ^d	
ls -l | awk '{if ($1 ~ /d/) print $0}'	
ls -dlr */	Prints in detail all non hidden folders in the current working directory in reverse alphabetical order.
ls -lr | grep ^d	
ls -lr | awk '{if ($1 ~ /d/) print $0}'	
ls -dt */	Prints all non hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -dtr */	Prints all non hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -dlt */	Prints in detail all non hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -lt | grep ^d	
ls -lt | awk '{if ($1 ~ /d/) print $0}'	
ls -dltr */	Prints in detail all non hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -ltr | grep ^d	
ls -ltr | awk '{if ($1 ~ /d/) print $0}'	
ls -di */	Prints all non hidden folders in the current working directory, including inode numbers, in alphabetical order.
ls -dri */	Prints all non hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dli */	Prints in detail all non hidden folders in the current working directory, including inode numbers, in alphabetical order.
ls -drli */	Prints in detail all non hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dti */	Prints all non hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtri */	Prints all non hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -dlti */	Prints in detail all non hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dltri */	Prints in detail all non hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

Hidden folders

Command	Output
ls -d .*/	Prints all hidden folders in the current working directory in alphabetical order.
ls -dr .*/	Prints all hidden folders in the current working directory in reverse alphabetical order.
ls -dl .*/	Prints in detail all hidden folders in the current working directory in alphabetical order.
ls -dlr .*/	Prints in detail all hidden folders in the current working directory in reverse alphabetical order.
ls -dt .*/	Prints all hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -dtr .*/	Prints all hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -dlt .*/	Prints in detail all hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -dltr .*/	Prints in detail all hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -di .*/	Prints all hidden folders in the current working directory, including inode numbers, in alphabetical order.
ls -dri .*/	Prints all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dli .*/	Prints in detail all hidden folders in the current working directory, including inode numbers, in alphabetical order.
ls -drli .*/	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dti .*/	Prints all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtri .*/	Prints all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

ls -dlti .*/	Prints in detail all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dltri .*/	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

Non hidden and hidden folders

Command	Output
ls -d */ .*/	Prints all non hidden and hidden folders in the current working directory in alphabetical order.
ls -dr */ .*/	Prints all non hidden and hidden folders in the current working directory in reverse alphabetical order.
ls -dl */ .*/	Prints in detail all non hidden and hidden folders in the current working directory in alphabetical order.
ls -dlr */ .*/	Prints in detail all non hidden and hidden folders in the current working directory in reverse alphabetical order.
ls -dt */ .*/	Prints all non hidden and hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -dtr */ .*/	Prints all non hidden and hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -dlt */ .*/	Prints in detail all non hidden and hidden folders in the current working directory in chronological order, going from newest to oldest.
ls -dltr */ .*/	Prints in detail all non hidden and hidden folders in the current working directory in reverse chronological order, going from oldest to newest.
ls -di */ .*/	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in alphabetical order.

ls -dri */ .*/	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dli */ .*/	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in alphabetical order.
ls -drli */ .*/	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dti */ .*/	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtri */ .*/	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -dlti */ .*/	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dltri */ .*/	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

B. Listing Files

Non hidden files

Command	Output
ls -p | grep -v /	Prints all non hidden files in the current working directory in alphabetical order.
ls -pr | grep -v /	Prints all non hidden files in the current working directory in reverse alphabetical order.
ls -pl | grep -v /	Prints in detail all non hidden files in the current working directory in alphabetical order.
ls -l | grep -v ^d	
ls -l | grep '^\-'	
ls -plr | grep -v /	Prints in detail all non hidden files in the current working directory in reverse alphabetical order.
ls -lr | grep -v ^d	
ls -lr | grep '^\-'	

ls -pt | grep -v /	Prints all non hidden files in the current working directory in chronological order, going from newest to oldest.
ls -ptr | grep -v /	Prints all non hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -plt | grep -v /	Prints in detail all non hidden files in the current working directory in chronological order, going from newest to oldest.
ls -lt | grep -v ^d	
ls -lt | grep '^\-'	
ls -pltr | grep -v /	Prints in detail all non hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -ltr | grep -v ^d	
ls -lt | grep '^\-'	
ls -pi | grep -v /	Prints all non hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -pri | grep -v /	Prints all non hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -pli | grep -v /	Prints in detail all non hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -li | grep -v ^d	
ls -plri | grep -v /	Prints in detail all non hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -lri | grep -v ^d	
ls -pti | grep -v /	Prints all non hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -ptri | grep -v /	Prints all non hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -plti | grep -v /	Prints in detail all non hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -lti | grep -v ^d	
ls -pltri | grep -v /	Prints in detail all non hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -ltri | grep -v ^d	

Hidden Files

Command	Output
ls -d .?*	Prints all hidden files in the current working directory in alphabetical order.
ls -a | grep '^\.'	
ls -dr .?*	Prints all hidden files in the current working directory in reverse alphabetical order.
ls -ar | grep '^\.'	
ls -ld .?*	Prints in detail all hidden files in the current working directory in alphabetical order.
ls -ldr .?*	Prints in detail all hidden files in the current working directory in reverse alphabetical order.
ls -dt .?*	Prints all hidden files in the current working directory in chronological order, going from newest to oldest.
ls -at | grep '^\.'	
ls -dtr .?*	Prints all hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -atr | grep '^\.'	
ls -dlt .?*	Prints in detail all hidden files in the current working directory in chronological order, going from newest to oldest.
ls -dltr .?*	Prints in detail all hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -di .?*	Prints all hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -dir .?*	Prints all hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -ldi .?*	Prints in detail all hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -ldri .?*	Prints in detail all hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dti .?*	Prints all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtri .?*	Prints all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

ls -dlti .?*	Prints in detail all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dltri .?*	Prints in detail all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

Non hidden and hidden files

Command	Output
ls -pa | grep -v /	Prints all non hidden and hidden files in the current working directory in alphabetical order.
ls -pra | grep -v /	Prints all non hidden and hidden files in the current working directory in reverse alphabetical order.
ls -pla | grep -v /	Prints in detail all non hidden and hidden files in the current working directory in alphabetical order.
ls -la | grep -v ^d	
ls -la | grep '^\-'	
ls -prla | grep -v /	Prints in detail all non hidden and hidden files in the current working directory in reverse alphabetical order.
ls -rla | grep -v ^d	
ls -lra | grep '^\-'	
ls -pta | grep -v /	Prints all non hidden and hidden files in the current working directory in chronological order, going from newest to oldest.
ls -ptra | grep -v /	Prints all non hidden and hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -plta | grep -v /	Prints in detail all non hidden and hidden files in the current working directory in chronological order, going from newest to oldest.
ls -lta | grep -v ^d	
ls -lta | grep '^\-'	
ls -pltra | grep -v /	Prints in detail all non hidden and hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -ltra | grep -v ^d	
ls -ltra | grep '^\-'	
ls -pai | grep -v /	Prints all non hidden and hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -prai | grep -v /	Prints all non hidden and hidden files in the current working directory, including inode numbers, in reverse alphabetical order.

ls -plai | grep -v /	Prints in detail all non hidden and hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -prlai | grep -v /	Prints in detail all non hidden and hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -ptia | grep -v /	Prints all non hidden and hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -ptrai | grep -v /	Prints all non hidden and hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -pltai | grep -v /	Prints in detail all non hidden and hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -pltrai | grep -v /	Prints in detail all non hidden and hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

C. Listing Folders And Files

Non hidden folders and files

Command	Output
ls	Prints all non hidden folders and files in the current working directory in alphabetical order.
ls --group-directories-first	Prints all non hidden folders in the current working directory in alphabetical order, followed by all non hidden files in the current working directory in alphabetical order.
ls -r	Prints all non hidden folders and files in the current working directory in reverse alphabetical order.
ls -r --group-directories-first	Prints all non hidden folders in the current working directory in reverse alphabetical order, followed by all non hidden files in the current working directory in reverse alphabetical order.

ls -l	Prints in detail all non hidden folders and files in the current working directory in alphabetical order.
ls -l --group-directories-first	Prints in detail all non hidden folders in the current working directory in alphabetical order, followed by all non hidden files in the current working directory in alphabetical order.
ls -lr	Prints in detail all non hidden folders and files in the current working directory in reverse alphabetical order.
ls -lr --group-directories-first	Prints in detail all non hidden folders in the current working directory in reverse alphabetical order, followed by all non hidden files in the current working directory in reverse alphabetical order.
ls -t	Prints all non hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -t --group-directories-first	Prints all non hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all non hidden files in the current working directory in chronological order, going from newest to oldest.
ls -tr	Prints all non hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.
ls -tr --group-directories-first	Prints all non hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all non hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -lt	Prints in detail all non hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -lt --group-directories-first	Prints in detail all non hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all non hidden files in the current working directory in chronological order, going from newest to oldest.

ls -ltr	Prints in detail all non hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.
ls -ltr --group-directories-first	Prints in detail all non hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all non hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -i	Prints all non hidden folders and files in the current working directory, including inode numbers, in alphabetical order.
ls -i --group-directories-first	Prints all non hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all non hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -ri	Prints all non hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -ri --group-directories-first	Prints all non hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all non hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -li	Prints in detail all non hidden folders and files in the current working directory, including inode numbers, in alphabetical order.
ls -li --group-directories-first	Prints in detail all non hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all non hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -lri	Prints in detail all non hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -lri --group-directories-first	Prints in detail all non hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all non hidden files in the current working directory, including inode numbers, in reverse alphabetical order.

ls -ti	Prints all non hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -ti --group-directories-first	Prints all non hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all non hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -tri	Prints all non hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -tri --group-directories-first	Prints all non hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all non hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -lti	Prints in detail all non hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -lti --group-directories-first	Prints in detail all non hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all non hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -ltri	Prints in detail all non hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -ltri --group-directories-first	Prints in detail all non hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all non hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

Hidden folders and files

Command	Output
ls -d .*	Prints all hidden folders and files in the current working directory in alphabetical order.
ls -d .[^.]*	Prints all hidden folders and files in the current working directory in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -d .* --group-directories-first	Prints all hidden folders in the current working directory in alphabetical order, followed by all hidden files in the current working directory in alphabetical order.
ls -d .[^.]* --group-directories-first	Prints all hidden folders in the current working directory in alphabetical order, followed by all hidden files in the current working directory in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dr .*	Prints all hidden folders and files in the current working directory in reverse alphabetical order.
ls -dr .[^.]*	Prints all hidden folders and files in the current working directory in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dr .* --group-directories-first	Prints all hidden folders in the current working directory in reverse alphabetical order, followed by all hidden files in the current working directory in reverse alphabetical order.
ls -dr .[^.]* --group-directories-first	Prints all hidden folders in the current working directory in reverse alphabetical order, followed by all hidden files in the current working directory in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dl .*	Prints in detail all hidden folders and files in the current working directory in alphabetical order.
ls -dl .[^.]*	Prints in detail all hidden folders and files in the current working directory in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dl .* --group-directories-first	Prints in detail all hidden folders in the current working directory in alphabetical order, followed by all hidden files in the current working directory in alphabetical order.

ls -dl .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory in alphabetical order, followed by all hidden files in the current working directory in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dlr .*	Prints in detail all hidden folders and files in the current working directory in reverse alphabetical order.
ls -dlr .[^.]*	Prints in detail all hidden folders and files in the current working directory in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dlr .* --group-directories-first	Prints in detail all hidden folders in the current working directory in reverse alphabetical order, followed by all hidden files in the current working directory in reverse alphabetical order.
ls -dlr .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory in reverse alphabetical order, followed by all hidden files in the current working directory in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dt .*	Prints all hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -dt .[^.]*	Prints all hidden folders and files in the current working directory in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dt .* --group-directories-first	Prints all hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all hidden files in the current working directory in chronological order, going from newest to oldest.
ls -dt .[^.]* --group-directories-first	Prints all hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all hidden files in the current working directory in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtr .*	Prints all hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.

ls -dtr .[^.]*	Prints all hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtr .* --group-directories-first	Prints all hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -dtr .[^.]* --group-directories-first	Prints all hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtl .*	Prints in detail all hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -dtl .[^.]*	Prints in detail all hidden folders and files in the current working directory in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtl .* --group-directories-first	Prints in detail all hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all hidden files in the current working directory in chronological order, going from newest to oldest.
ls -dtl .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all hidden files in the current working directory in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtrl .*	Prints in detail all hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.
ls -dtrl .[^.]*	Prints in detail all hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.

ls -dtrl .* --group-directories-first	Prints in detail all hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -dtrl .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -di .*	Prints all hidden folders and files in the current working directory, including inode numbers, in alphabetical order.
ls -di .[^.]*	Prints all hidden folders and files in the current working directory, including inode numbers, in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -di .* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -di .[^.]* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dri .*	Prints all hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dri .[^.]*	Prints all hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dri .* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in reverse alphabetical order.

ls -dri .[^.]* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dli .*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in alphabetical order.
ls -dli .[^.]*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dli .* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -dli .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dlri .*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dlri .[^.]*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dlri .* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -dlri .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all hidden files in the current working directory, including inode numbers, in reverse alphabetical order. Returns an error if at least one hidden folder or at least one hidden file does not exist.

ls -dti .*	Prints all hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dti .[^.]*	Prints all hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dti .* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dti .[^.]* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtri .*	Prints all hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -dtri .[^.]*	Prints all hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtri .* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -dtri .[^.]* --group-directories-first	Prints all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.

ls -dtli .*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtli .[^.]*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtli .* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -dtli .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtrli .*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -dtrli .[^.]*	Prints in detail all hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.
ls -dtrli .* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

ls -dtrli .[^.]* --group-directories-first	Prints in detail all hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest. Returns an error if at least one hidden folder or at least one hidden file does not exist.

Non hidden and hidden folders and files

Command	Output
ls -a	Prints all non hidden and hidden folders and files in the current working directory in alphabetical order.
ls -a --group-directories-first	Prints all non hidden and hidden folders in the current working directory in alphabetical order, followed by all non hidden and hidden files in the current working directory in alphabetical order.
ls -ar	Prints all non hidden and hidden folders and files in the current working directory in reverse alphabetical order.
ls -ar --group-directories-first	Prints all non hidden and hidden folders in the current working directory in reverse alphabetical order, followed by all non hidden and hidden files in the current working directory in reverse alphabetical order.
ls -la	Prints in detail all non hidden and hidden folders and files in the current working directory in alphabetical order.
ls -la --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory in alphabetical order, followed by all non hidden and hidden files in the current working directory in alphabetical order.
ls -lra	Prints in detail all non hidden and hidden folders and files in the current working directory in reverse alphabetical order.
ls -lra --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory in reverse alphabetical order, followed by all non hidden and hidden files in the current working directory in reverse alphabetical order.

ls -ta	Prints all non hidden and hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -ta --group-directories-first	Prints all non hidden and hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all non hidden and hidden files in the current working directory in chronological order, going from newest to oldest.
ls -tra	Prints all non hidden and hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.
ls -tra --group-directories-first	Prints all non hidden and hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all non hidden and hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -tla	Prints in detail all non hidden and hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -tla --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory in chronological order, going from newest to oldest, followed by all non hidden and hidden files in the current working directory in chronological order, going from newest to oldest.
ls -trla	Prints in detail all non hidden and hidden folders and files in the current working directory in reverse chronological order, going from oldest to newest.
ls -trla --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory in reverse chronological order, going from oldest to newest, followed by all non hidden and hidden files in the current working directory in reverse chronological order, going from oldest to newest.
ls -ia	Prints all non hidden and hidden folders and files in the current working directory, including inode numbers, in alphabetical order.

ls -ia --group-directories-first	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all non hidden and hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -iar	Prints all non hidden and hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -iar --group-directories-first	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all non hidden and hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -ial	Prints in detail all non hidden and hidden folders and files in the current working directory, including inode numbers, in alphabetical order.
ls -ial --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in alphabetical order, followed by all non hidden and hidden files in the current working directory, including inode numbers, in alphabetical order.
ls -lrai	Prints in detail all non hidden and hidden folders and files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -lrai --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in reverse alphabetical order, followed by all non hidden and hidden files in the current working directory, including inode numbers, in reverse alphabetical order.
ls -tia	Prints all non hidden and hidden folders and files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -tia --group-directories-first	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all non hidden and hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -tiar	Prints all non hidden and hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

ls -tiar --group-directories-first	Prints all non hidden and hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all non hidden and hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -tlia	Prints in detail all non hidden and hidden folders and files in the current working directory in chronological order, going from newest to oldest.
ls -tlia --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in chronological order, going from newest to oldest, followed by all non hidden and hidden files in the current working directory, including inode numbers, in chronological order, going from newest to oldest.
ls -tlari	Prints in detail all non hidden and hidden folders and files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.
ls -tlari --group-directories-first	Prints in detail all non hidden and hidden folders in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest, followed by all non hidden and hidden files in the current working directory, including inode numbers, in reverse chronological order, going from oldest to newest.

Sources:

https://stackoverflow.com/questions/14352290/listing-only-directories-using-ls-in-bash-an-examination

https://serverfault.com/questions/368370/how-do-i-exclude-directories-when-listing-files

https://www.cyberciti.biz/faq/bash-shell-display-only-hidden-dot-files/

https://askubuntu.com/questions/468901/how-to-show-only-hidden-files-in-terminal

Imran Afzal for the command: ls -a | grep '^\.'

LEFT CLICK TO RESTART FROM THE TOP