جامعة القاهرة الجديدة التكنولوجية

Fall 2023

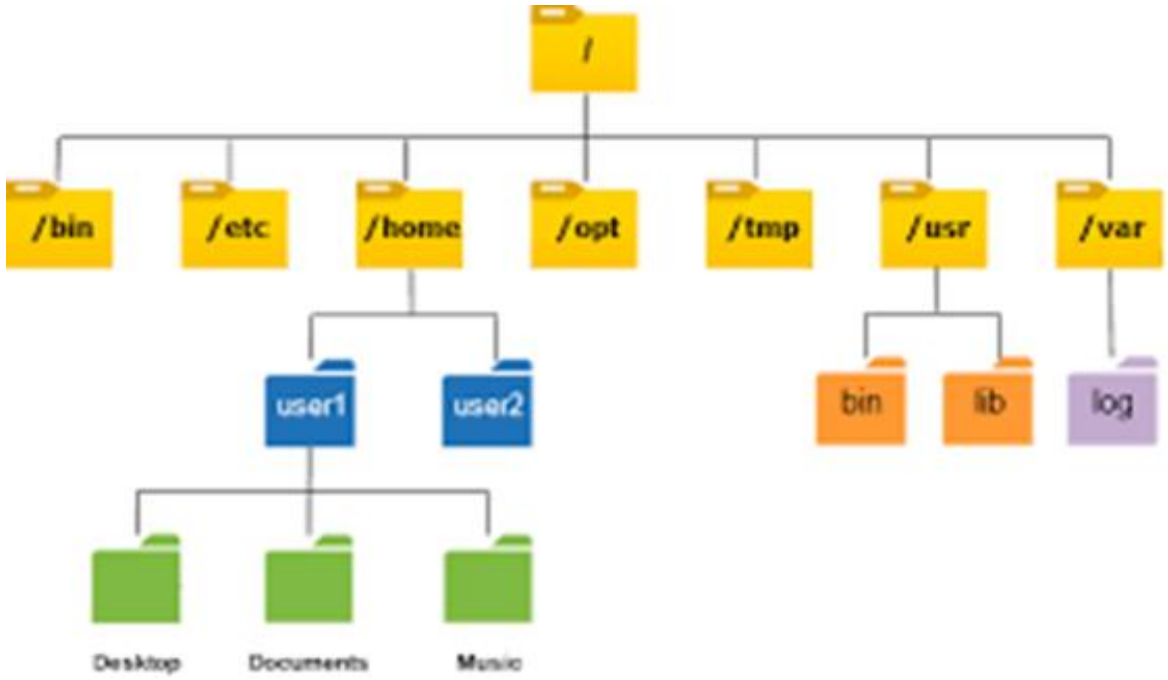
(7)

Linux Essentials

Dr. Hatem Yousry

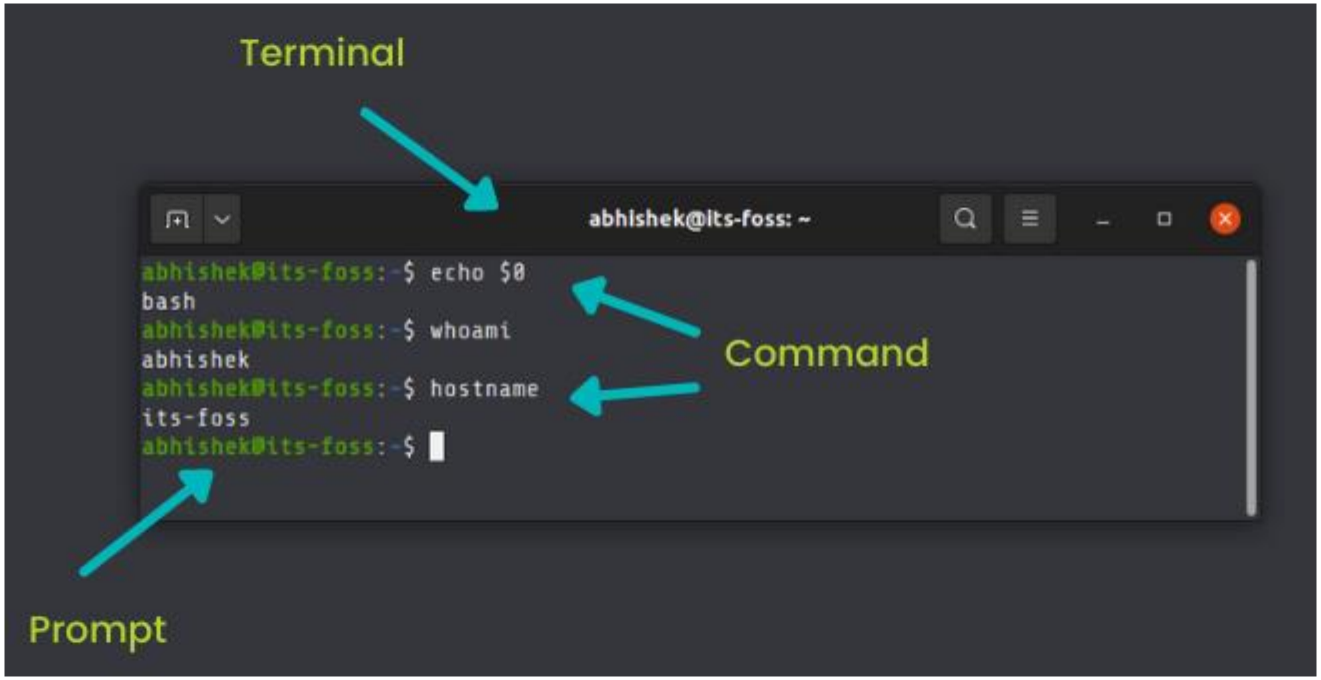
Agenda

- Directory Tree.
- Directory Commands.



User Interface and dialog utility

- Good program/shell script must interact with users.
- There are two ways to this one is **use command line** to script when you want input, second **use statement like echo and read** to read input into variable from the prompt.



```
abhishek@its-foss:~$ echo $0
bash
abhishek@its-foss:~$ whoami
abhishek
abhishek@its-foss:~$ hostname
its-foss
abhishek@its-foss:~$
```

- **$ cat > userinte**
- **#**
- **# Script to demo echo and read command for user interaction**
- **#**
- **echo "Your good name please :"**
- **read na**
- **echo "Your age please :"**
- **read age**
- **neyr=`expr $age + 1`**
- **echo "Hello $na, next year you will be $neyr yrs old."**

- Save it and run as
- **$ chmod +x userinte**
- **$ ./userinte**

```
Your good name please :
Hatem
Your age please :
25
Hello Hatem, next year you will be 26 yrs old.
```

File and Path Names

- In Linux file names, you are essentially allowed to use any character that your computer can display (and then some).
- An easy trap for beginners to fall into is the fact that Linux distinguishes uppercase and lowercase letters in file names.
- Unlike Windows, where uppercase and lowercase letters in file names are displayed but treated the same (**case sensitive**), Linux considers **x- files and X- Files two different file names.**
- A further difference from DOS and Windows computers is that Linux **does not use suffixes** to characterize a file's "type". Hence, the **dot** is a completely ordinary character within a file name. You are free to store a text as mumble.txt, but mumble would be just as acceptable in principle. This should of course not turn you off using suffixes completely—you do after all make it easier to identify the file content.

Hidden files

- As another peculiarity, file names starting with a dot (**“.”**) will be skipped in some places, for example when the files within a directory are listed—files with such names are considered **“hidden”.**
- This feature is often used for files containing settings for programs and which should not distract users from more important files in directory listings.
- For DOS and Windows experts: These systems allow “hiding” files by means of a **“file attribute”** which can be set independently of the file’s name. Linux and Unix do not support such a thing.

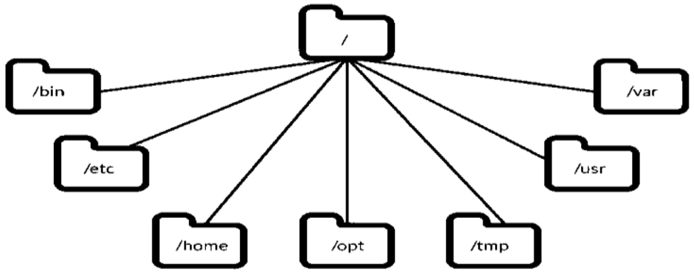
Linux file names

- Some programs insist on their input files having specific suffixes. The C compiler, gcc, for example, considers files with names ending in **“.c”** C source code, those ending in **“.s “** assembly language source code, and those ending in **“.o”** precompiled object files.
- File names should always start with one of the letters or a digit:
- X-files
- foo.txt.bak
- 50.something
- 7_of_9
- On the contrary, problems would be possible:
- -10°F *Starts with “- ”,* includes special character
- .profile Will be hidden
- 3/4-metre Contains illegal character
- Smörrebröd Contains umlauts

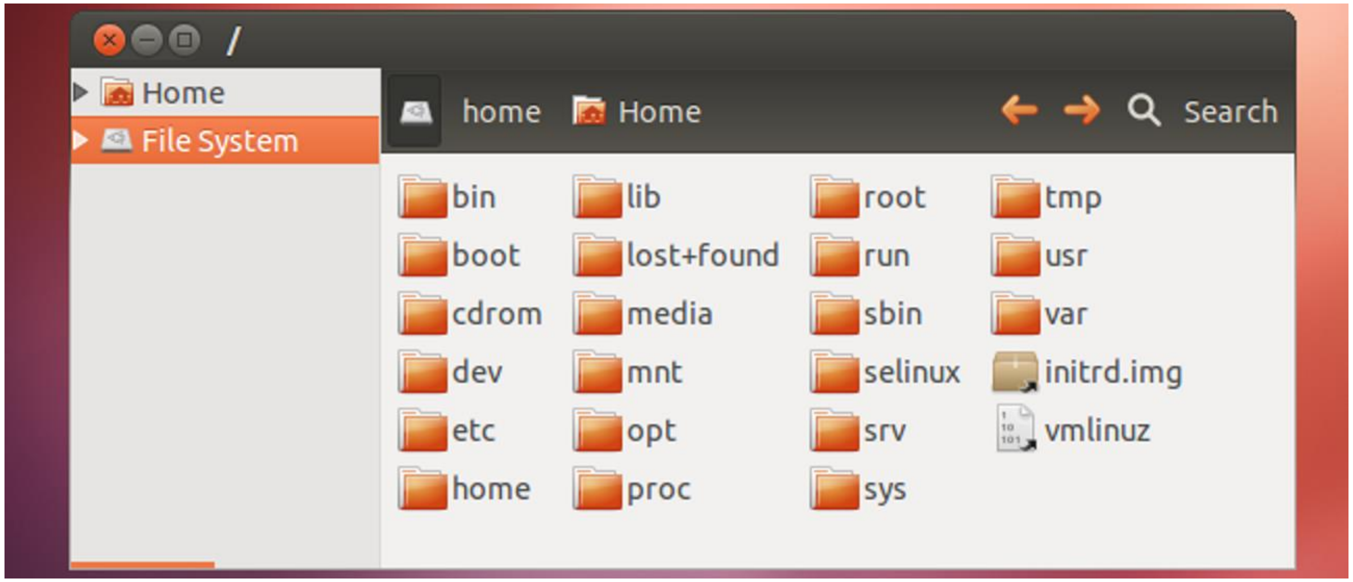
Directories

- Since potentially many users may work on the same Linux system, it would be problematic if **each file name could occur just once**. It would be difficult to make clear to user Joe that he cannot create a file called letter.txt since user Sue already has a file by that name.
- In addition, there must be a (convenient) way of ensuring that Joe cannot read all of Sue's files and the other way round.
- For this reason, Linux supports the idea of hierarchical **"directories"** which are used to group files.
- **File names do not need to be unique within the whole system**, but only within the same directory. This means in particular that the system can assign **different directories** to Joe and Sue, and that within those they may call their files whatever they please without having to worry about each other's files.

Directory Tree



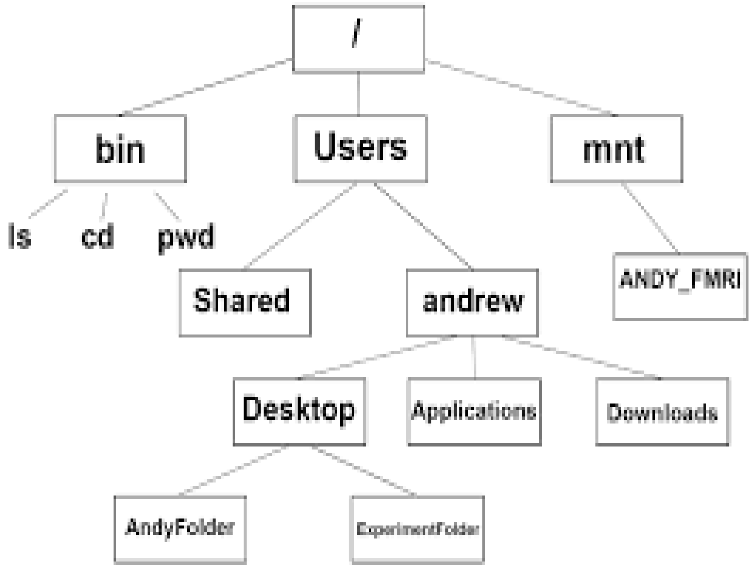
- Directories may contain other directories (this is the term **“hierarchical” or tree** which results in a tree-like structure (inventively called **a “directory tree”).**
- A Linux system has a special directory which forms the root of the tree and is therefore called the “root directory”.
- **Its name is “/” (slash).**

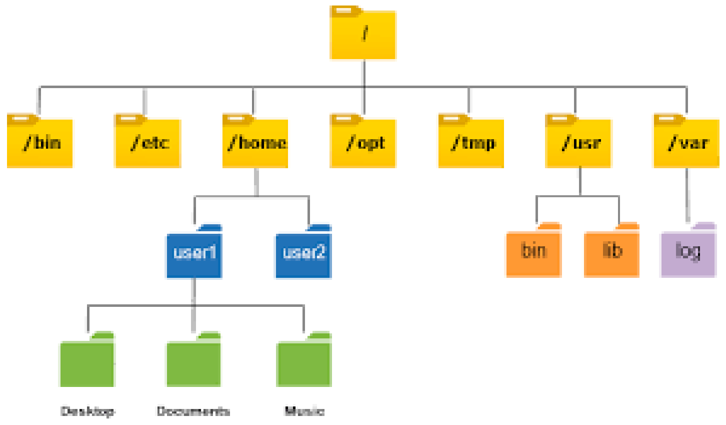


bin	lib	root	tmp
boot	lost+found	run	usr
cdrom	media	sbin	var
dev	mnt	selinux	initrd.img
etc	opt	srv	vmlinuz
home	proc	sys	

The Linux Directory Structure

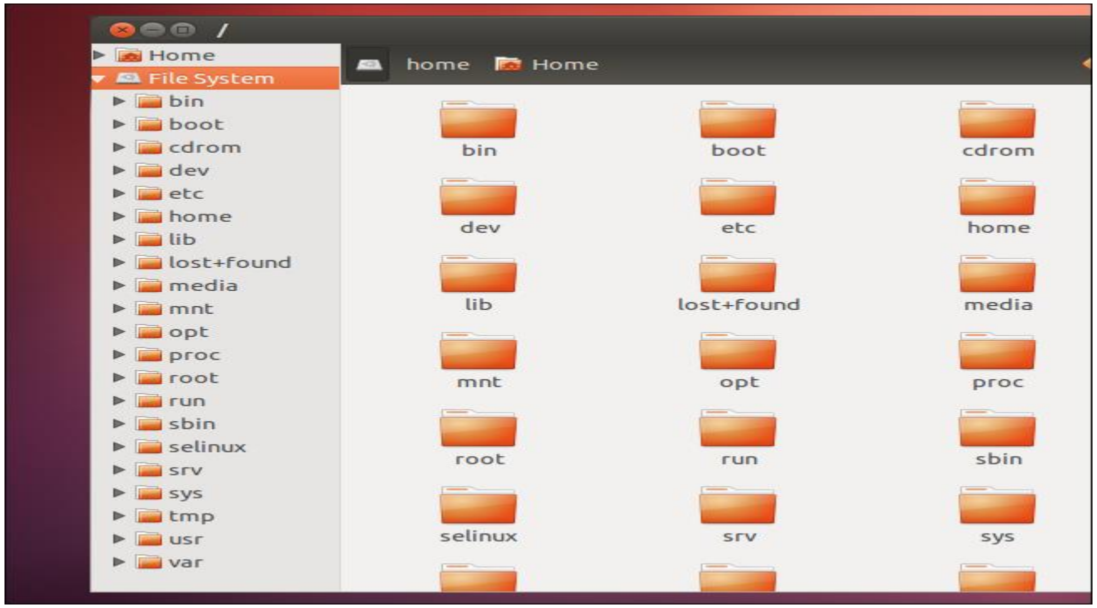
- / — The Root Directory.
- /bin — Essential User Binaries.
- /boot — Static Boot Files.
- /cdrom — Historical Mount Point for CD-ROMs. ...
- /dev — Device Files.
- /etc — Configuration Files.
- /home — Home Folders.
- /lib — Essential Shared Libraries.





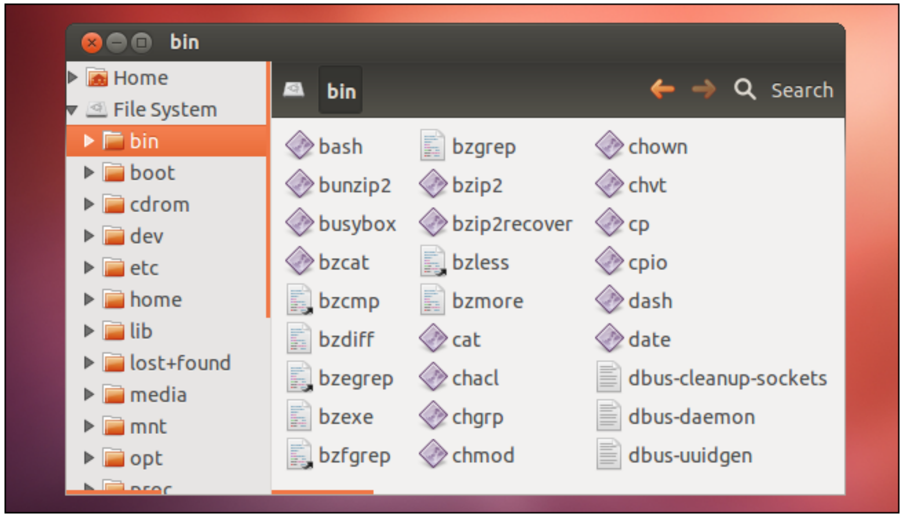
/ — The Root Directory

- Everything on your Linux system is located under the / directory, known as the root directory. You can think of the / directory as being similar to the **C:\ directory** on Windows — but this isn't strictly true, as **Linux doesn't have drive letters.** While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.



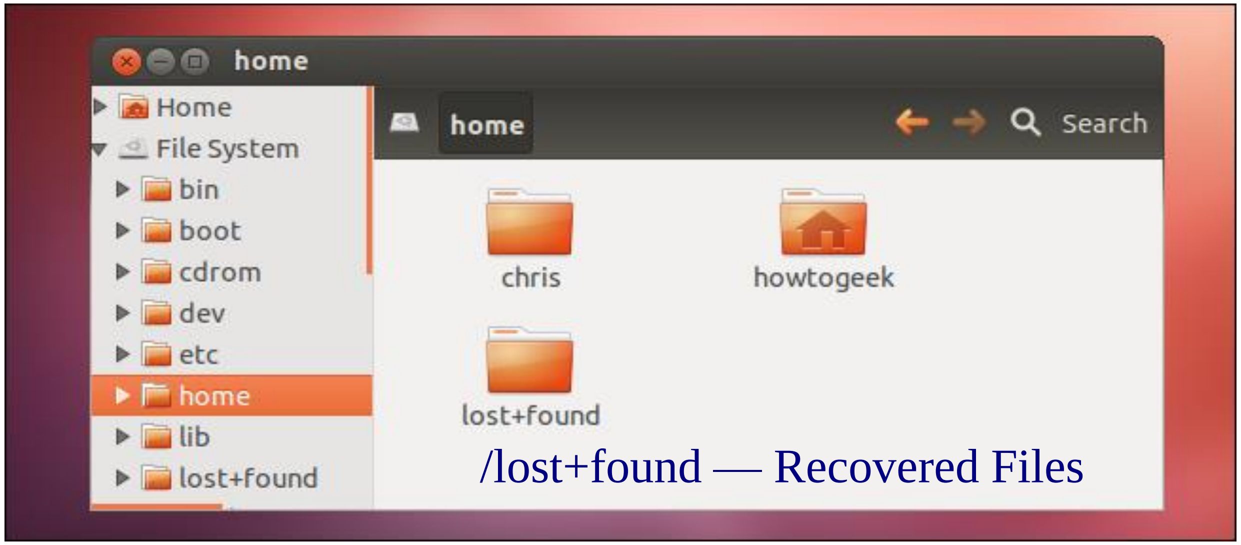
/bin — Essential User Binaries

- The /bin directory contains the **essential user binaries (programs)** that must be present when the system is mounted in single-user mode. Applications such as Firefox are stored in /usr/bin, while important system programs and utilities such as the bash shell are located in /bin. The /usr directory may be stored on another partition — placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted. The /sbin directory is similar — it contains essential system administration binaries.



/home — Home Folders

- The /home directory contains a home folder for each user.
- For example, if your user name is bob, you have a home folder located at /home/bob. This home folder **contains the user's data files and user-specific configuration files.** Each user only has write access to their own home folder and must obtain elevated permissions (become the root user) to modify other files on the system.



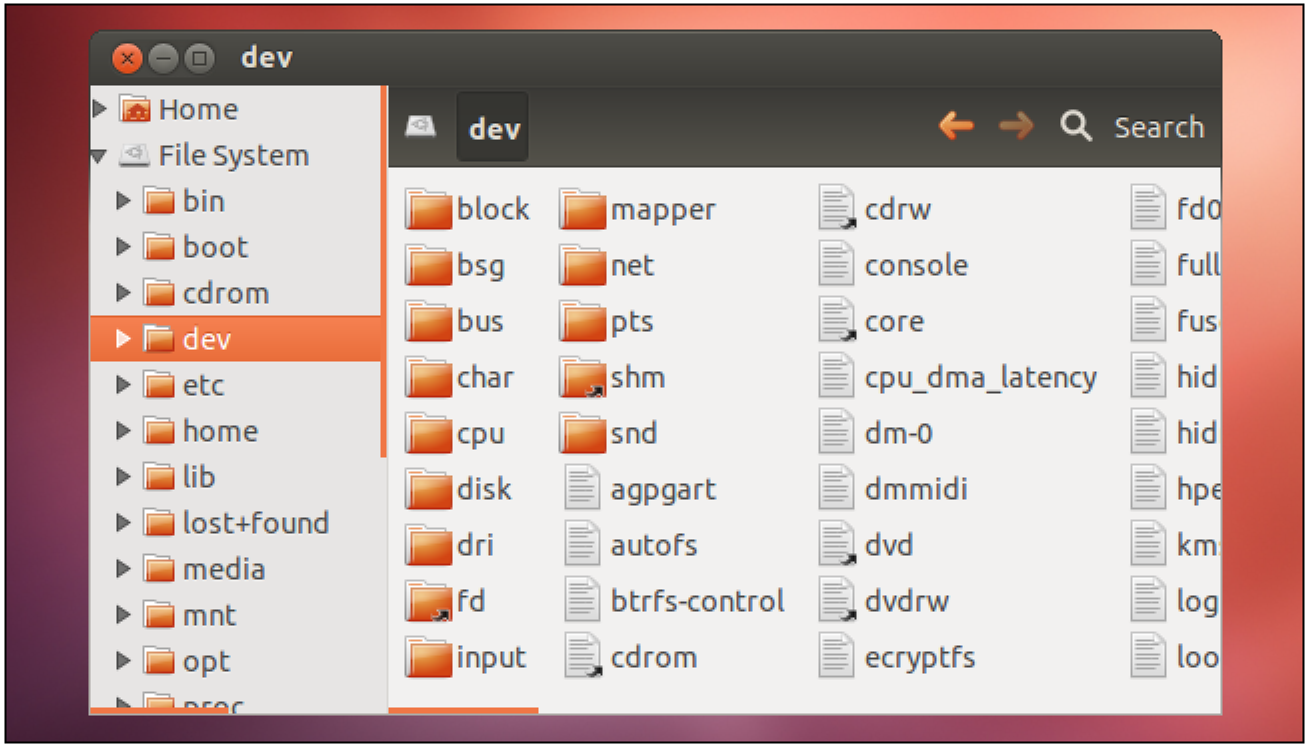
/lost+found — Recovered Files

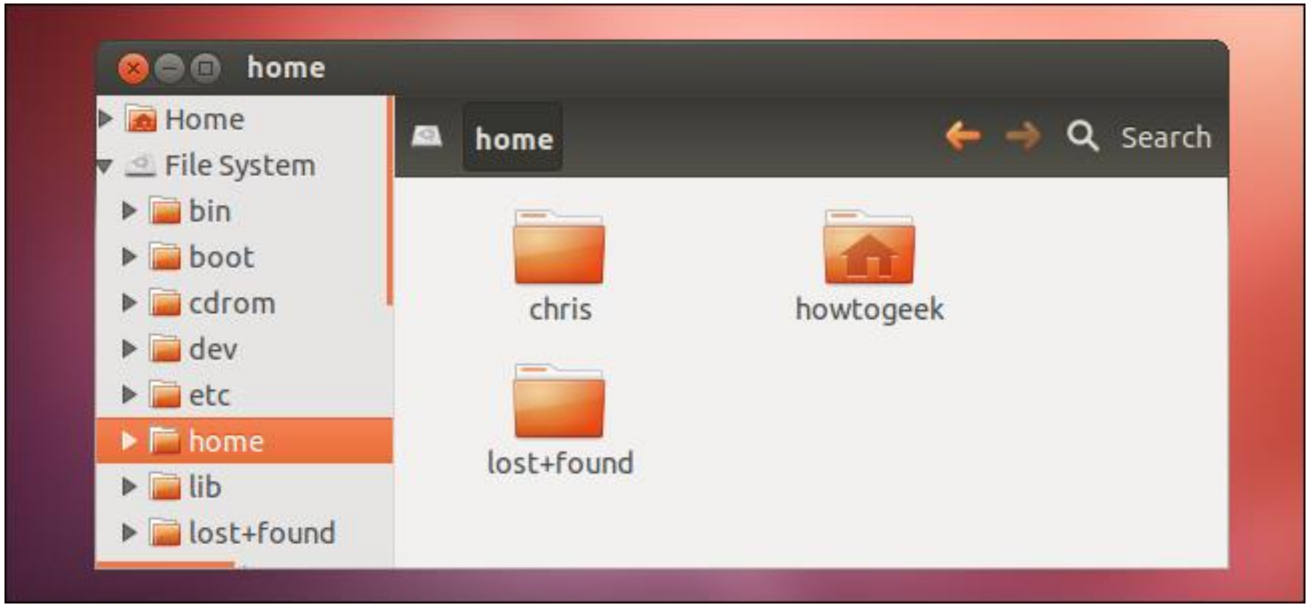
/boot - /cdrom

- **/boot — Static Boot Files**
- The /boot directory contains the files needed to boot the system — for example, the GRUB **boot loader's** files and your Linux kernels are stored here. The boot loader's configuration files aren't located here, though — they're in /etc with the other configuration files.
- **/cdrom — Historical Mount Point for CD-ROMs**
- The /cdrom directory isn't part of the FHS standard, but you'll still find it on Ubuntu and other operating systems. It's a **temporary location for CD-ROMs** inserted in the system. However, the standard location for temporary media is inside the /media directory.

/dev — Device Files

- **/dev — Device Files**
- Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices. These are not actual files as we know them, but they appear as files — for example, **/dev/sda represents the first SATA drive in the system.** If you wanted to partition it, you could start a partition editor and tell it to edit /dev/sda.
- This directory also contains **pseudo-devices**, which are **virtual devices** that don't actually correspond to hardware. For example, /dev/random produces random numbers. **/dev/null** is a **special device that produces no output and automatically discards all input** — when you pipe the output of a command to /dev/null, you discard it.


dev
Home
File System
bin
boot
cdrom
dev
etc
home
lib
lost+found
media
mnt
opt
proc
dev
Search
block
bsg
bus
char
cpu
disk
dri
fd
input
mapper
net
pts
shm
snd
agpgart
autofs
btrfs-control
cdrom
cdrw
console
core
cpu_dma_latency
dm-0
dmmidi
dvd
dvdrw
ecryptfs
fd0
full
fus
hid
hid
hpe
km
log
loo


home
Home
File System
bin
boot
cdrom
dev
etc
home
lib
lost+found
home
Search
chris
howtogeek
lost+found

/lost+found - /etc

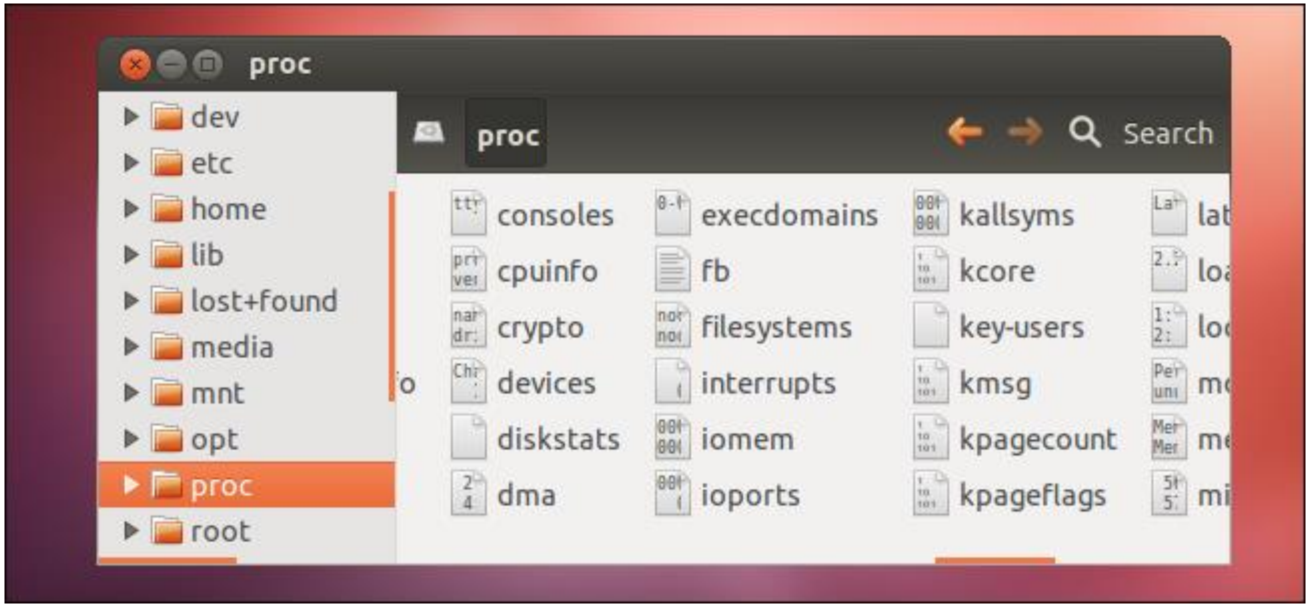
- **/lost+found — Recovered Files**
- Each Linux file system has a lost+found directory. If the file system crashes, a file system check will be performed at **next boot**. Any corrupted files found will be placed in the lost+found directory, so you can **attempt to recover as much data as possible.**
- **/etc — Configuration Files**
- The /etc directory contains **configuration files**, which can generally be edited by hand in a text editor. Note that the /etc/ directory contains system-wide configuration files — **user-specific configuration** files are located in each user's home directory.

/lib - /media

- **/lib — Essential Shared Libraries**
- The /lib directory contains **libraries** needed by the essential binaries in the /bin and /sbin folder. Libraries needed by the binaries in the /usr/bin folder are located in /usr/lib.
- **/media — Removable Media**
- The /media directory contains subdirectories where removable media devices inserted into the computer are mounted. For example, when you insert a **CD into your Linux system**, a directory will automatically be created inside the /media directory. **You can access the contents of the CD inside this directory.**

/mnt - /opt - /proc

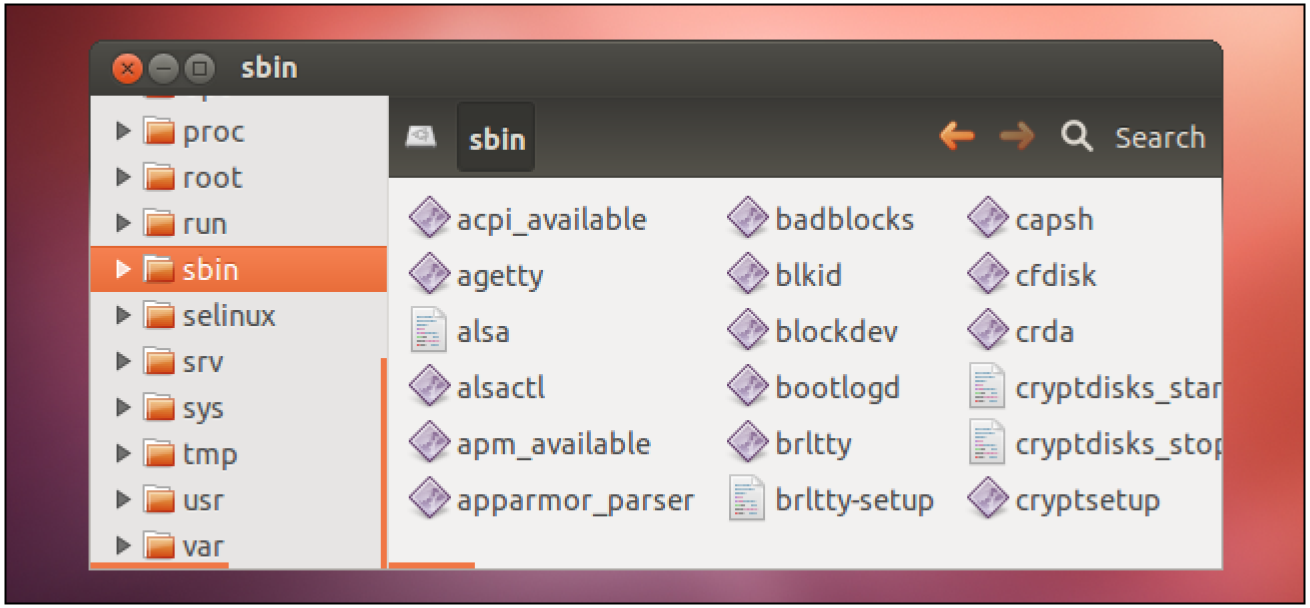
- **/mnt — Temporary Mount Points**
- Historically speaking, the /mnt directory is where system **administrators mounted temporary** file systems while using them. For example, if you're **mounting a Windows partition** to perform some file recovery operations, you might mount it at /mnt/windows. However, you can mount other file systems anywhere on the system.
- **/opt — Optional Packages**
- The /opt directory contains subdirectories for **optional software packages.** It's commonly used by proprietary software that doesn't obey the standard file system hierarchy — for example, a proprietary program might dump its files in /opt/application when you install it.
- **/proc — Kernel & Process Files**
- The /proc directory similar to the /dev directory because it doesn't contain standard files. It contains special files that **represent system and process information.**


proc

dev
etc
home
lib
lost+found
media
mnt
opt
proc
root

proc
Search

consoles
cpuinfo
crypto
devices
diskstats
dma
execdomains
fb
filesystems
interrupts
iomem
ioports
kallsyms
kcore
key-users
kmsg
kpagecount
kpageflags
lat
loa
loc
mo
me
mi


sbin

proc
root
run
sbin
selinux
srv
sys
tmp
usr
var

sbin
Search

acpi_available
agetty
alsa
alsactl
apm_available
apparmor_parser
badblocks
blkid
blockdev
bootlogd
brltty
brltty-setup
capsh
cfdisk
crda
cryptdisks_star
cryptdisks_stop
cryptsetup

/root - /run - /sbin

- **/root — Root Home Directory**
- The /root directory is the home directory of the **root user**. Instead of being located at /home/root, it's located at /root. This is distinct from /, which is the system root directory.
- **/run — Application State Files**
- The /run directory is fairly new, and gives applications a standard place to **store transient files** they require like sockets and process IDs. These files can't be stored in /tmp because files in /tmp may be deleted.
- **/sbin — System Administration Binaries**
- The /sbin directory is similar to the /bin directory. It contains essential binaries that are generally intended to **be run by the root user for system administration.**

/selinux - /srv

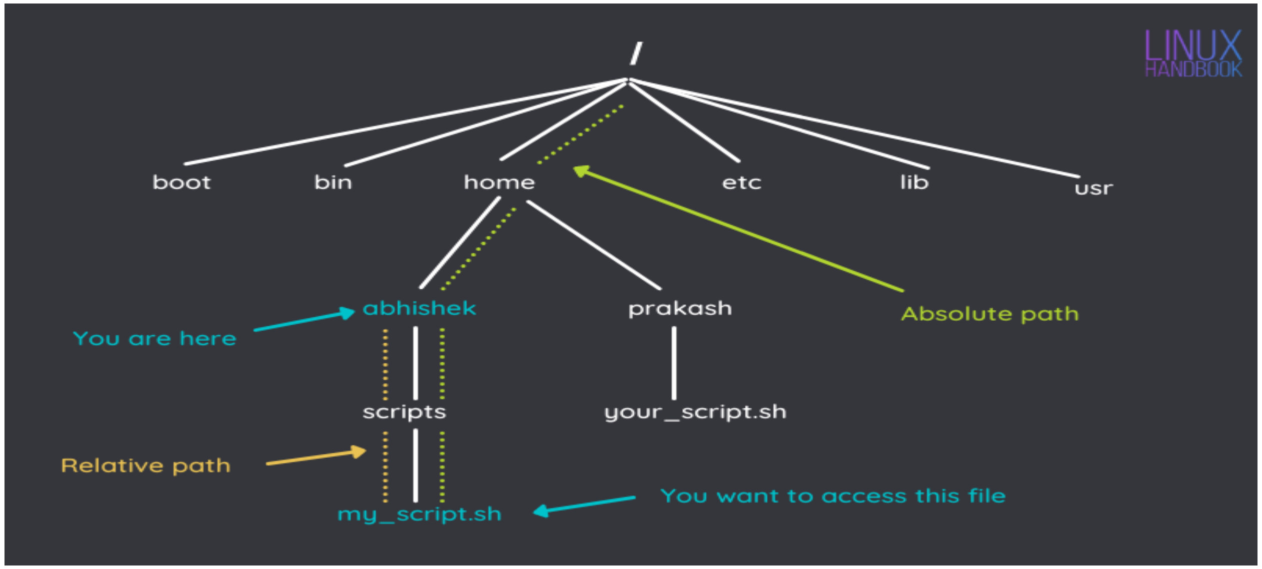
- **/selinux — SELinux Virtual File System**
- If your Linux distribution uses SELinux for **security** (Fedora and Red Hat, for example), the /selinux directory contains special files used by SELinux. It's similar to /proc. **Ubuntu doesn't use SELinux,** so the presence of this folder on Ubuntu appears to be a bug.
- **/srv — Service Data**
- The /srv directory contains **"data for services provided by the system."** If you were using the Apache HTTP server to serve a website, you'd likely **store your website's files** in a directory inside the /srv directory.

/tmp - /usr

- **/tmp — Temporary Files**
- **Applications store temporary files** in the /tmp directory. These files are generally deleted whenever your system is restarted and may be deleted at any time by utilities such as tmpwatch.
- **/usr — User Binaries & Read-Only Data**
- The /usr directory contains **applications and files used by users**, as opposed to applications and files used by the system. For example, **non-essential applications** are located inside the /usr/bin directory instead of the /bin directory and non-essential system administration binaries are located in the /usr/sbin directory instead of the /sbin directory. Libraries for each are located inside the /usr/lib directory. The /usr directory also contains other directories — for example, architecture-independent files like graphics are located in /usr/share.

Absolute and Relative Path Names

- Every file in a Linux system is described by a name which is constructed by starting at the root directory and mentioning every directory down along the path to the one containing the file, followed by the name of the file itself.
- For example, **/home/joe/letter.txt** names the file letter.txt, which is located within the joe directory, which in turn is located within the home directory, which in turn is a direct descendant of the root directory.

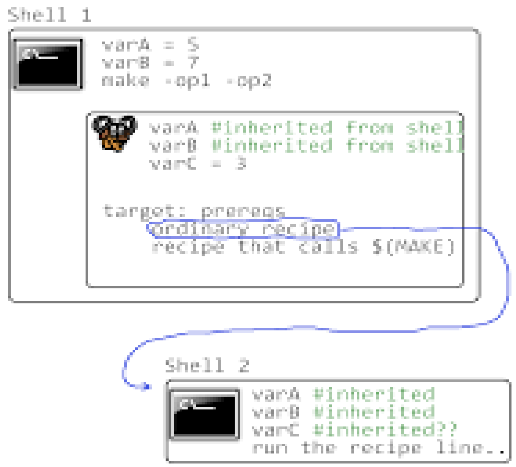


Absolute and Relative Path Names

- A name that starts with the root directory is called an **"absolute path name"**—we talk about "path names" since the name describes a "path" through the directory tree, which may contain directory and file names (i. e., it is a collective term).
- Each process within a Linux system has a "current directory" (often also called **"working directory"**).
- File names are searched within this directory; letter.txt is thus a convenient abbreviation for "the file called letter.txt in the current directory", and sue/letter.txt stands for "the file letter.txt within the sue directory within the current directory". Such names, which start from the current directory, are called **"relative path names".**
- **A path name starting with a "/" is absolute; all others are relative.**

Shell Inheritance

- The current directory is “inherited” between parent and child processes. So if you start a new shell (or any program) from a shell, **that new shell uses the same current directory as the shell you used to start it.**
- In your new shell, you can change into another directory using the **cd command**, but the current directory of the old shell does not change—if you leave the new shell, you are back to the (unchanged) current directory of the old shell.



```
varA = 5
varB = 7
make -op1 -op2

varA #inherited from shell
varB #inherited from shell
varC = 3

target: prereqs
    ordinary recipe
    recipe that calls $(MAKE)
```

```
varA #inherited
varB #inherited
varC #inherited??
run the recipe line..
```

Directory Commands

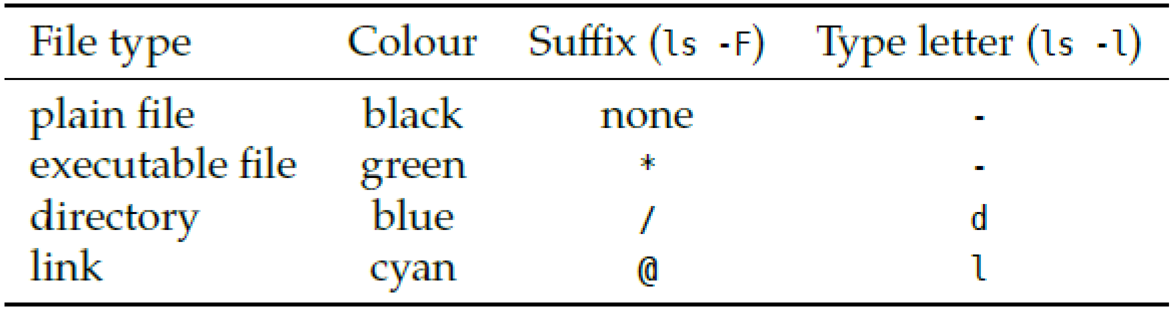
- The Current Directory: cd & Co.
- You can use the **cd** shell command to change the current directory: Simply give the desired directory as a parameter:
- **$ cd letters** Change to the letters directory
- **$ cd ..** Change to the directory above
- If you do not give a parameter you will end up in your home directory:
- **$ cd**
- **$ pwd**
- /home/joe
- You can output the absolute path name of the current directory using the **pwd** current directory (“print working directory”) command.

The tilde (“~”) & The “cd -”

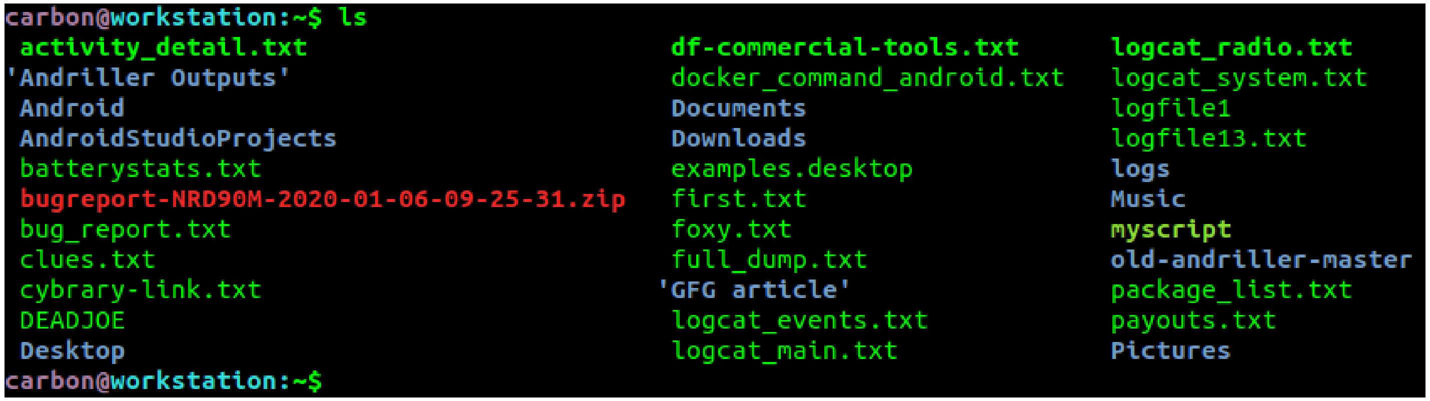
- Possibly you can also see the current directory as part of your prompt: Depending on your system settings there might be something like
- joe@red:~/letters> _
- where **~/letters** is short for **/home/joe/letters**; the tilde **(“~”)** stands for the current user’s home directory.
- The **“cd -”** command changes to the directory that used to be current before the most recent cd command. This makes it convenient to alternate between two directories.

Listing Files and Directories—ls

- To find one's way around the directory tree, it is important to be able to find out which files and directories are located within a directory. The ls ("list") command does this.
- Some file type designations in ls



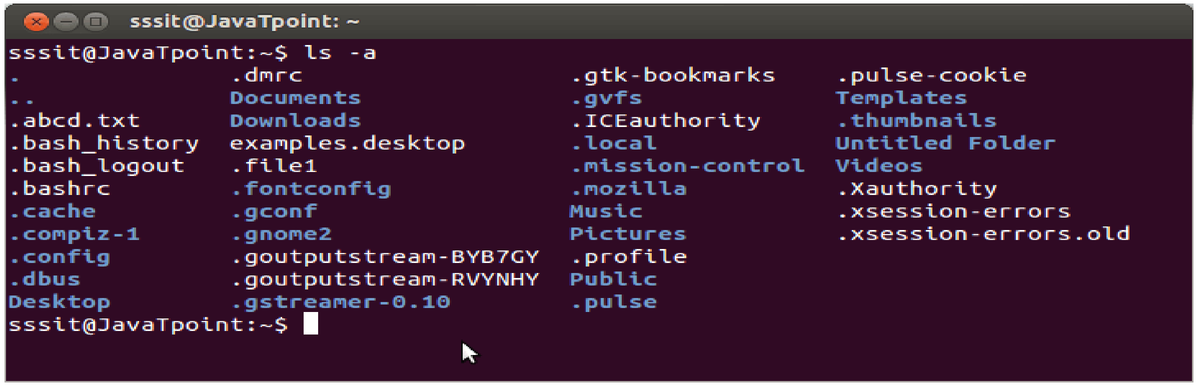
File type	Colour	Suffix (`ls -F`)	Type letter (`ls -l`)
plain file	black	none	-
executable file	green	*	-
directory	blue	/	d
link	cyan	@	l



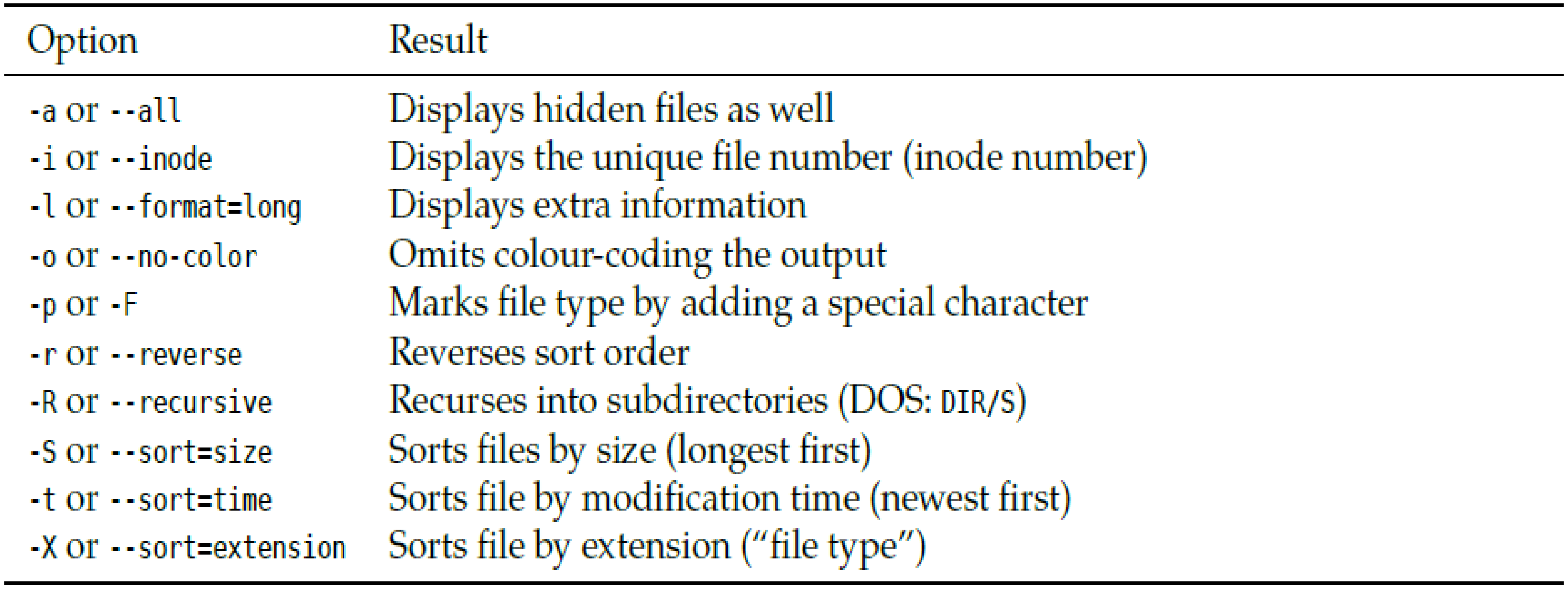
```
carbon@workstation:~$ ls
 activity_detail.txt                      df-commercial-tools.txt      logcat_radio.txt
'Andriller Outputs'                       docker_command_android.txt   logcat_system.txt
 Android                                  Documents                    logfile1
 AndroidStudioProjects                    Downloads                    logfile13.txt
 batterystats.txt                         examples.desktop             logs
 bugreport-NRD90M-2020-01-06-09-25-31.zip first.txt                    Music
 bug_report.txt                           foxy.txt                     myscript
 clues.txt                                full_dump.txt                old-andriller-master
 cybrary-link.txt                        'GFG article'                 package_list.txt
 DEADJOE                                  logcat_events.txt            payouts.txt
 Desktop                                  logcat_main.txt              Pictures
carbon@workstation:~$
```

Some ls options

- **ls -r** :It is used to print the list in reverse order.
- **ls -R** :It will display the content of the sub-directories also.
- **ls –lX** :It will group the files with same extensions together in the list.
- **ls -lt** :It will sort the list by displaying recently modified filed at top.
- **ls ~** :It gives the contents of home directory.
- **ls ../** :It give the contents of parent directory.
- **ls -a** :In Linux, hidden files start with . (dot) symbol and they are not visible in the regular directory. The (ls -a) command will enlist the whole list of the current directory including the hidden files.


```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ ls -a
.                .dmrc                  .gtk-bookmarks     .pulse-cookie
..               Documents              .gvfs              Templates
.abcd.txt        Downloads              .ICEauthority      .thumbnails
.bash_history    examples.desktop       .local             Untitled Folder
.bash_logout     .file1                 .mission-control   Videos
.bashrc          .fontconfig            .mozilla           .Xauthority
.cache           .gconf                 Music              .xsession-errors
.compiz-1        .gnome2                Pictures           .xsession-errors.old
.config          .goutputstream-BYB7GY  .profile
.dbus            .goutputstream-RVYNHY  Public
Desktop          .gstreamer-0.10        .pulse
sssit@JavaTpoint:~$
```

Some ls options



Option	Result
`-a` or `--all`	Displays hidden files as well
`-i` or `--inode`	Displays the unique file number (inode number)
`-l` or `--format=long`	Displays extra information
`-o` or `--no-color`	Omits colour-coding the output
`-p` or `-F`	Marks file type by adding a special character
`-r` or `--reverse`	Reverses sort order
`-R` or `--recursive`	Recurses into subdirectories (DOS: `DIR/S`)
`-S` or `--sort=size`	Sorts files by size (longest first)
`-t` or `--sort=time`	Sorts file by modification time (newest first)
`-X` or `--sort=extension`	Sorts file by extension (“file type”)

- **$ ls**
- file.txt
- file2.dat
- **$ ls –l**
- -rw-r--r-- 1 joe users 4711 Oct 4 11:11 file.txt
- -rw-r--r-- 1 joe users 333 Oct 2 13:21 file2.dat

Creating and Deleting Directories: mkdir and rmdir

- To create new directories, the mkdir command is available. **It requires one or more directory names as arguments,** otherwise you will only obtain an error message instead of a new directory. To create nested directories in a single step, you can use the -p option, otherwise the command assumes that all directories in a path name except the last one already exist.
- For example:
- $ **mkdir pictures/holiday**
- mkdir: cannot create directory `pictures/holiday': No such file
- or directory
- $ **mkdir -p pictures/holiday**
- $ **cd pictures**
- $ **ls -F**
- holiday/

Creating and Deleting Directories: mkdir and rmdir

- you can remove it using the rmdir (“remove directory”) command.
- As with mkdir, at least one path name of a directory to be deleted must be given.
- In addition, the directories in question must be empty, i. e., they may not contain entries for files, subdirectories, etc.
- Again, only the last directory in every name will be removed:
- **$ rmdir pictures/holiday**
- **$ ls -F**

- pictures/

Creating and Deleting Directories: mkdir and rmdir

- With the -p option, **all empty subdirectories mentioned in a name can be removed in one step**, beginning with the one on the very right.
- **$ mkdir -p pictures/holiday/summer**
- **$ rmdir pictures/holiday/summer**
- **$ ls -F pictures**
- pictures/holiday/
- **$ rmdir -p pictures/holiday**
- **$ ls -F pictures**
- ls: pictures: No such file or directory

Copying Files and Directories (cp)

- cp file1 file2 is the command which makes a copy of file1 in the current working directory and calls it file2.
- What we are going to do now is to take a file stored in an open access area of the file system, and use the cp command to copy it to your unixstuff directory.
- First, change to your unixstuff directory.
- **cd ~/unixstuff**
- Then at the shell prompt type: **cp/cm/shared/training/tutorial/science.txt .**
- Directories can also be copied with the **cp** command, but it's necessary to add the option **–R** to do so. This option means 'recursive' and will copy the contents of the directory as well as the directory itself, for example: **cp -R directory1 directory2**
- **cp -R /cm/shared/training/tutorial ~/unixstuff**

Moving files and Directories (mv)

- The move command has a variety of similar but subtly different uses. It can be used to move a file to a different location (i.e. a different directory). It can also be used to move multiple files to a different directory. It can also be used to rename a file or a directory.
- For example: **mv file1 directory1/**
- This would move file1 from the current directory into directory1.
- **mv file1 file2 file3 directory1/**
- This would move file1, file2 and file3 from the current directory into directory1. mv file1 file2
- This would rename file1 as file2. mv directory1/ directory2/
- This would rename a directory.
- Finally, **mv file1 directory/file2**
- This would move and rename a file in one step.

Removing Files (rm) and Directories (rmdir)

- To delete (remove) a file, use the rm command.
- As an example, we are going to create a copy of the science.txt file then delete it.
- Inside your unixstuff directory, type **cp science.txt tempfile.txt ls rm tempfile.txt ls**
- In order to delete an empty directory you can use the command rmdir directory
- However this won't remove directories that already have files in them, instead you can use **rm -r directory** to recursively delete files in directory (use sparingly - there is no Recycle bin!)
- You can use the rmdir command to remove a directory (make sure it is empty first).
- Try to remove the backups directory. You will not be able to since Linux will not let you remove a non-empty directory.

Thank You

Linux

Dr. Hatem Yousry
E-mail: Hyousry@nctu.edu.eg