Basics

>>Unix is a name given to a family of operating systems like ubuntu, Redhat linux, hp-ux, aix etc. most of the programs written for one os is NOT compatible with other OS. Non-unix family OS example is Windows.

>> Shell is a program which is used to start other programs. Knowledge about shell is needed to automate commonly used tasks, and for the Unix system administration.

>>If the shell offers the facility to read its commands from a text file, then the aforementioned syntax and features may be thought of as a programming language, and such files may be thought of as scripts

>>Unix architecture is "Terminal" based OS. ie, many users using the SAME computer with their own keyboard and monitor. where as in the case of PC the computer has only one keyboard and monitor.

>> some unix terminals capable of displaying GUI, known as x-terminals.

>> Unlike Windows, unix has no concept of file type.

>> The program "file" is used to make a guess as to a file's contents
>> files whose names start with a dot "." are considered hidden. ls -a will display hidden files.

>> "." is current directory and ".." is a parent directory. these are treated as hidden.

>> who am i --> displays the user name
>> which command-name --> this gives path name of the program.
ex: which bash
>> id --> displays the uid and gid of the user.

>> su username --> temporarily run the shell as another user.

>> To change the PS1 or terminal prompt to something useful.
edit .bashrc or .profile depends on the Operating system details.

Adding users:

>> sudo adduser -g <group> <username>
sudo passwd <username> --> it prompts you type the new password.
sudo usermod -g <newgroup> <username>

>> "id" this gives the current user, group details.
>> "groups" command gives the current user is part of group names.

>> adduser <username>

adduser [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID] [--firstuid ID] [--lastuid ID] [--gecos GECOS] [--ingroup GROUP | --gid ID]

[--disabled-password] [--disabled-login] [--encrypt-home] USER
   Add a normal user

adduser --system [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID]
[--gecos GECOS] [--group | --ingroup GROUP | --gid ID] [--disabled-password]
[--disabled-login] USER
   Add a system user

adduser --group [--gid ID] GROUP
addgroup [--gid ID] GROUP
   Add a user group

addgroup --system [--gid ID] GROUP
   Add a system group

adduser USER GROUP
   Add an existing user to an existing group

++

Getting help in Unix

>> ls --help
>> man ls
>> whatis command gives partial help.
whatis ls output: *ls (1) - list directory contents*
>> file <some_filename> → this gives the type of the file like text file or binary file or etc.
file home.tar output: *home.tar: tar archive*

++

Aliases

>> Various useful aliases can be created for the administration purpose. Refer the "Unix for Oracle DBAs Pocket Reference.pdf"
Aliases are great tool to create the shortcuts, some of them are below.

>> After entering the aliases in .bashrc or .profile based on the OS, execute . ~/.bashrc in your terminal (there should be a space between . and ~/.bashrc)

>> alias nuke_java="ps -ef | grep java | awk '{print $2}'|xargs kill -9"
>> alias was_home="cd /opt/IBM/WebSphere/Appserver"
>> alias profile_home="cd /opt/IBM/WebSphere/Appserver/profiles"
>> alias logs="cd $profile_home/logs"

Aliases are also used in avoiding some of the pitfalls of unix
>> alias rm="rm -i"   --> it will make the rm as interactive.

>> alias nuke_java="ps -ef | grep java | awk '{print $2}' | xargs kill -9" --> we can type nuke_java to kill all java processes

++

ls-cd-rm-mkdir-cp-mv

Ls

Ref: https://www.garron.me/en/go2linux/ls-file-permissions.html
>> ls -r --> see the listing in reverse order of alphabetically.
>> **ls -C** --> arranges files alphabetically in columns.
>> To list the files in the directory, "ls" is not that useful, you can use find command.
find . -type f
>> ls -ld --> is very useful command, it will (the 'd' flag) give one line output about the file or directory information. the 'l' flag will long list the files in the directory.
ex. output.: -rwxrwxr-x 1 root root 2907 Jun 15 2002 ./news/newsbot/old/3filter.pl

>> **ls -d */** --> to list all sub directories in the directory.

>> d- directory; l - link.
>> ls -lh --> list out based on the sizes in human readable.
>> find . -maxdepth 1 -type f --> this will list only files in the current directory.
>> Least recently touched files:

>> ls -alt |head --> to see the recently touched files.touched means, either the file is read or written recently. -a: all, -l:list out the touched time, -t: sort by touch time.

```
total 248
drwxr-xr-x 25 vivek vivek  4096 Feb  2 20:05 .
-rw-r--r--  1 vivek vivek  3672 Feb  2 20:05 .bashrc
drwxrwxr-x  3 vivek vivek  4096 Feb  2 20:04 bin
-rw-------  1 vivek vivek  5918 Feb  2 13:08 .xsession-errors
-rw-------  1 vivek vivek  6643 Feb  2 13:03 .bash_history
-rw-------  1 vivek vivek   155 Feb  2 13:03 .Xauthority
-rw-rw-r--  1 vivek vivek     0 Feb  1 22:56 zerolengthfile
drwxrwxr-x  3 vivek vivek  4096 Feb  1 22:42 .cinnamon
drwx------  3 vivek vivek  4096 Feb  1 22:42 .gconf
```

>> List recently changed files: **ls -alc | sort -rk9**
-c: lists the recently changed files.
sort -r: reverse

>> **find . -type f -perm -664** -->list files which have rw rw r permissions

>> ls -R --> this shows the all the files in all the directories of current location

++

Cd – change directory

>> Tilde ~ represents the home directory for ex: /home/vivek
>> cd ~ --> cd to /home/vivek
>> cd - --> this will take you to the previous directory.

++

Mkdir

mkdir -p /target/dirctory

++

Rm

>> to delete a folder, then use "rm -rf dir_name"
>> another way is : find dir_name -mindepth 1 -delete
>> or "find dir_name -mindepth 1 -exec rm -rf {}"
>> delete a folder and then recreate it: rm -rf dir_name && mkdir dir_name
>> rm -f filename --> "f" indicates force.
>> alias rm="rm -i" --> this aliias will make the rm interactive. alias can be added to the .bashrc or .profile file. and then run ". ./.bashrc" f (dot space dot slash) to take the effect.

++


cp.pdf

++

Filters:

>> filters take input, does some processing on the input and produce o/p. wc is a filter which takes standard input (either from keyboard or from anohter FILE using < or from another PROGRAM using |)

>> filter examples.
cat -- does nothing to the input
more -- paginate the input
grep --> removes the lines of input which does not contain the specified string.
wc --> counts lines, words, and characters.
sort --> sorts the input.
sed --> basic editing
awk --> any editing you name it.
Tee → reads standard input and sends the o/p to screen and files.

Ex: **ls | tee –a file1 file2** → a will append the content to the file1 and file2. And the o/p will be displayed on the screen and written to both file1 and file2.
++

Less

less filename
>> 'F' to act as "tail -f" type mode.
>> 'G' to go to the end of the file, g to go to top
>> /search-string --> to search for a string. n will go the next result and N will take you to the previous result,
>> ? for search in backward direction
>> 'q' to quit the less command.
>>

++

find

Ref:http://alvinalexander.com/unix/edu/examples/find.shtml

>> GREP is for finding out the string pattern inside a file. FIND is an utility for searching file and folders based on size, access time, modification time. **The basic difference between find & grep is that FIND is for searching file names and directories at system level while GREP is for searching a pattern inside a file.**

>> **find . -name "host*" -type f -maxdepth 1**

This above cammond will find the all the file names which starts with "host. "-maxdepth 1" switch will search only in the current directory only. "-type f" will search for files only, "-type d" will search directories. (find in the current directory (dot means current directory), -name swith tells what is the name that you are looking for and -maxdepth 1 tells search in the current directory only.)

>> **find /users/al -name Cookbook -type d** (find for Cookbook directory(-type d) in the "/users/al" directory.
>> **find /opt /usr /var -name foo.scala -type f** (you can search in multiple directories)
>> **find . -type f -not -name "*.html"** (find files that does NOT match the pattern of html)
>> **find . -type f \(-name "*.c" -o -name "*.sh" \)(**Its AND operator. find both .c and .sh files)
>> **find . -name "*.conf" -exec grep -li "Display" {} \;** (This will find the conf files in the current directory, which has the string "Display" in them)
>> **find . -name "*.conf" -exec ls -ld {} \;** (This will list out the file locations and their details)
>> **find . -type d -maxdepth 1 -exec ls -ld {} \;** (This will display only directories and their details)
>> find . -type d -maxdepth 1 |xargs ls –ld (same as above, just using xargs here)
>> ls -al | grep '^d' → another way of displaying only directories. ^d means lines start with d.
>> ls -d */ → to list all the sub-directories in the dir. Same as above.
>> ls -al | grep '^-' → find only files in the current directory.
>> **find . -type f -name "Foo*" -exec rm {} \;** (This will find files with Foo* and deletes them)
>> **find . -iname foo** (case insensitive find)
>> **find . -mtime -7 -type f** (this will find the files that were modiefied (-mtime) in the last 7 days.)

++

Grep

At the end of the document there are Reference examples are there.
>> **sudo find /etc -type f -exec grep -li "vivek" {} \;** --> this is used to find the string "vivek" in the etc folder. "l" will list the name of the file in which it found the string. "i" will ignore the case.
>> quote around the search string is not needed when you are not specifying regular expression.
>> **grep -n "started" SytemOut.log** --> this will show the line number of the matched search in the SytemOut.log file.
>> **grep -B5 -A10 "e-business" SytemOut.log** --> it will show 5 lines before and 10 lines after the SystemOut.log
>> **ls -al | grep "\s[.\w+]"** --> this will find out all the files started with .
>> **grep -v "vivek" /etc/passwd** --> NOT. find the lines in the file which does not have the "vivek" string.
>> **cat sample.txt | grep -c string** --> this will count the number of occurrences the String appears in the sample.txt
>> **netstat -ant |grep -iE 'listen|estab*'** → grep for either of the 2 strings. Flag “i” is to ignore case “E” is to extended grep.
>> **grep -e pattern1 -e pattern2 filename**

Reference Examples:
---
search for a string in one or more files

```
---------------------------------------
grep 'fred' /etc/passwd              # search for lines containing 'fred'
in /etc/passwd
grep fred /etc/passwd                # quotes usually not when you don't
use regex patterns
grep null *.scala                    # search multiple files

case-insensitive
----------------
grep -i joe users.txt                # find joe, Joe, JOe, JOE, etc.

regular expressions
-------------------
grep '^fred' /etc/passwd             # find 'fred', but only at the start
of a line
grep '[FG]oo' *                      # find Foo or Goo in all files in
the current dir
grep '[0-9][0-9][0-9]' *             # find all lines in all files in the
current dir with three numbers in a row

display matching filenames, not lines
-------------------------------------
grep -l StartInterval *.plist        # show all filenames containing the
string 'StartInterval'
grep -il StartInterval *.plist       # same thing, case-insensitive

show matching line numbers
--------------------------
grep -n we gettysburg-address.txt    # show line numbers as well as the
matching lines

lines before and after grep match
---------------------------------
grep -B5 "the living" gettysburg-address.txt        # show all matches,
and five lines before each match
grep -A10 "the living" gettysburg-address.txt       # show all matches,
and ten lines after each match
grep -B5 -A5 "the living" gettysburg-address.txt    # five lines before
and ten lines after

reverse the meaning
-------------------
grep -v fred /etc/passwd             # find any line *not* containing
'fred'
grep -vi fred /etc/passwd            # same thing, case-insensitive

grep in a pipeline
------------------
ps auxwww | grep httpd               # all processes containing 'httpd'
ps auxwww | grep -i java             # all processes containing 'java',
ignoring case
ls -al | grep '^d'                   # list all dirs in the current dir

search for multiple patterns
----------------------------
egrep 'apple|banana|orange' *
# search for multiple patterns, all files in current dir
```

```
egrep -i 'apple|banana|orange' *
# same thing, case-insensitive
egrep 'score|nation|liberty|equal' gettysburg-address.txt
# all lines matching multiple patterns
locate -i calendar | grep Users | egrep -vi 'twiki|gif|shtml|drupal-
7|java|PNG'       # oh yeah
       (see http://alvinalexander.com/linux-unix/linux-egrep-multiple-
regular-expres...)

multiple search strings, multiple filename patterns
-----------------------------------------------------
grep -li "jtable" $(find . -name "*.java,v" -exec grep -li "prevayl" {}
\;)
find all files named "*.java,v" containing both 'prevayl' and 'jtable'

grep + find
-----------
find . -type f -exec grep -il 'foo' {} \;     # print all filenames of
files under current dir containing 'foo', case-insensitive

recursive grep search
---------------------
grep -rl 'null' .                             # very similar to the
previous find command; does a recursive search
grep -ril 'null' /home/al/sarah /var/www      # search multiple dirs
egrep -ril 'aja|alvin' .                      # multiple patterns,
recursive
      (see http://alvinalexander.com/linux-unix/recursive-grep-r-
searching-egrep-find)

grep gzip files
---------------
zgrep foo myfile.gz                           # all lines containing the
pattern 'foo'
zgrep 'GET /blog' access_log.gz               # all lines containing 'GET
/blog'
zgrep 'GET /blog' access_log.gz | more        # same thing, case-
insensitive


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

grep AND OR examples.pdf

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Sort

```
Ref:http://www.skorks.com/2010/05/sort-files-like-a-master-with-the-
linux-sort-command-bash/
-------------
```

>> **ls -al | sort -nk5** --> sort the 5th colum(k5) and "n" means numerical.
>> **ps -ef | sort -rnk2** --> "r" reverse the sort. n - numerical . k2 - 2nd colum.
>> **sort files > files.sorted** --> sort actully does not change the file content. it just displays the sorted file. if you want to save the sorted file, you need to redirect the sorted view to anothe file.
>> sort -n numbers.txt --> sort the file based on numerical value.
>> Any blank character acts as a column separator for the sort program. This is default behaviour. But we can specify column separator with "t" flag.
>> "**k" flag tells the column number.**
>> **cat /etc/passwd | head | sort -t: -k1** --> we are sorting the /etc/passwd file based on the first column and we specified that : is the column separator.
>> sort -t. -k1,1n -k4,4n ips.txt --> (-k1,1n) it tells that sort by 1st column(k1) to 1st column(1n)-numerically. (-k4,4n): 4th column to 4th column numerically.

>> **sort file1 > file1 --> Never do this**. It will erase the file1. The shell assumes that, whatever may be the output of the command, first it needs to prepare the output file. in this case it is file1. If this file is already existed, it will be erased and if not new file will be created. so when you "cat file1" after the above sort command, nothing will be displayed. if you want to store a sorted file, execute the below command.
>> sort file1 > file2
mv file2 file1
++

Locate

>> locate tomcat.sh
it is used to search the file names in the entire system of unix. It is a simpler vesion of "find" command.

++

SED

sed: stream editor. It is a filter that sits on the way of data stream and changes the data as per the action. It just displays the changed file or output content on the screen, but does NOT change the file.
>> These are the 3 ways of executing the sed.
sed actions [files]
sed -e action1 -e action2 [files]
sed -f action-file [files]
>> the most common action is text substitution (find- replace). the actions is: 's/foo/bar/' (here it replaces first foo with bar in each line of the entire file.)
>> **sed -e 's/india/Bharata/' testfile.**txt --> this will replace india to Bharata in each line first match.
>> **sed -e 's/india/Bharata/g' testfile.txt** --> this will replace ALL matches of india to Bharata.
>> Another common action is deleting the lines.
sed -e '/india/d' testfile.txt --> it will delete all the lines with india in it.

>> **sed -e '/india/!d' testfile.txt** --> here the "!d" causes not to delete the lines which has india in it. all other lines will be deleted.
>> **sed '1,10d' testfile** --> this will delete lines from 1 to 10.
>> **sed '90,$d'** testfile --> this will delete the lines from 90 to the end of the file.
>> **sed '90,$!d' testfile** --> this will NOT delete the lines from 90 to the end of the file, ie, it will delete lines from 1 to 90.
>> we can write executable sed scripts just like shell scripts. Example below

```
#! /bin/sed -f
s/india/Bharat/
/america/d
```

save this file with any name for ex: sedscript
chmod +x sedscript --> this will make the script executable.

```
>> cat testfile | sedscript
```

Sed scripts can be used to remove the control-m characters. (to type ^M, press control and press v and m)

```
>> sed -e "s/^M//" file_whihc_has_ctrl_M > newfilename
>> mv newfilenamme original filename
```

++

awk:

awk is grand daddy of all text processing filters.
>> $1, $2 etc represents the first column, 2nd column of the output of a piping program like who or ls -l etc. or they are the tokens from the each line of the input to awk program.
>> **who | awk '{print $1,"is the name."}'**
>> **ps -ef | awk '{print $8}'** --> it will print the names of the processes.
>> **ps -ef | grep java |awk '{print $1}'| xargs kill -9** --> this will kill the java processes.
>> by default the token seperator is space or tab. but we can explictity tell what the separator is with -F option.
>> awk -F : '{print $1,"home:",$6}' /etc/passwd
>> awk -f scriptfile inputfile --> the awk commands in scriptfile and the inputfile has the data on which the awk commands works.
>> awk -F: '{printf("%-12s%20s\n",$1,$6)}' /etc/passwd --> the first token will be left justfied and it will be embedded in 12 palce holders and the 4th token will be embedded in 20 placeholders and it is right justfied. see the below output.

```
root                                            /root
daemon                                      /usr/sbin
bin                                              /bin
sys                                              /dev
sync                                             /bin
mail                                    /var/spool/mail
```

>>
++

File:

file command tells what type of the file it is.
file /etc/passwd --> output is /etc/passwd: ASCII text

++

Symbolic Links and Hard Links:

>> Links and symbolic links are different in unix. In Unix, it is possible for a file to have more than one name. "ln" is used to create hard link or just a link. No link is considered more important than any other link. Their inodes are same.

>> Symbolic link is created using **"ln -s"**. A small file which contains the reference to the original file will be created. It is more like a shortcut in Windows. Inodes are different for the original file and the symbolic link.

>> ls -l will list the symbolic link with "l" in the listing.


links-symbolic and hard-reference.pdf

++

typeset

typeset is used to force the variable to either lower case or upper case
ex:
typeset -l var1 --> var1 has been set to lowercase (-u for uppercase)
var1=LSKDHFLKSDHF
echo $var1
o/p: lskdhflksdhf

++

Inode & stat:

>> df -i --> gives you the percentage of Inodes used in each disk volumes.
>> ls -i --> displays the inode numbers of any file in some unix flavors. In some ls -i will work.
>> **stat /user/lib** (or any directory or any volume /dev/sda1)
>> find . -inum 38988 -exec rm {} \; --> to remove file which has inode number 38988. {} will contain the file name.
>> find . -inum 38988 -exec mv {} file_new \;

>> In unix everything is a file. Directory is also a file. A directory in unix is just a file with file names and their inode numbers.

++

jobs & fg:

>> to list the jobs which were stopped (via ctrl+z), "jobs" command will give the list
>> To resume the stopped jobs, "fg" command can be used.
>> jobs which were running asynchronously using "&" were can not be stopped.
>> "nice" command is used to give priority to the processes. -19 highest priority, 10- defaulut priority, +19 is lowest priority.
>>

++

Nohup:

>> Most of the time you login into remote server via ssh. If you start a shell script or command and you exit (abort remote connection), the process / command will get killed. Sometime job or command takes a long time. If you are not sure when the job will finish, then it is better to leave job running in background. But, if you log out of the system, the job will be stopped and terminated by your shell. What do you do to keep job running in the background when process gets SIGHUP?

>> nohup command-name & --> you can run a backup command like this, so even you exit from the shell, the command will run asynchronously or in the background.
>> screen command also does the same.
>> a process can be run in the background or asynchronously just by using the "&", but the job will be terminated when the "logout" command was issued. so to continue, even after the logout, "nohup" command can be used. (nohup--no hangup)

++

Chmod-chown –chgrp


chmod.pdf


chown.pdf

++

Ctime-mtime-atime

http://www.unix.com/tips-and-tutorials/20526-mtime-ctime-atime.html

---

Unix keeps 3 timestamps for each file: **mtime, ctime, and atime.** Most people seem to understand atime (access time), it is when the file was last read. There does seem to be some confusion between mtime and ctime though. ctime is the inode change time while mtime is the file modification time. "Change" and "modification" are pretty much synonymous. There is no clue to be had by pondering those words. Instead you need to focus on what is being changed. **mtime changes when you write to the file. It is the age of the data in the file. Whenever mtime changes, so does ctime. But ctime changes a few extra times. For example, it will change if you change the owner or the permissions on the file**.

Let's look at a concrete example. We run a package called Samba that lets PC's access files. To change the Samba configuration, I just edit a file called smb.conf. (This changes mtime and ctime.) I don't need to take any other action to tell Samba that I changed that file. Every now and then Samba looks at the mtime on the file. If the mtime has changed, Samba rereads the file. Later that night our backup system runs. It uses ctime, which also changed so it backs up the file. But let's say that a couple of days later I notice that the permissions on smb.conf are 666. That's not good..anyone can edit the file. So I do a "chmod 644 smb.conf". This changes only ctime. Samba will not reread the file. But later that night, our backup program notices that ctime has changes, so it backs up the file. That way, if we lose the system and need to reload our backups, we get the new improved permission setting.

Here is a second example. Let's say that you have a data file called employees.txt which is a list of employees. And you have a program to print it out. The program not only prints the data, but it obtains the mtime and prints that too. Now someone has requested an employee list from the end of the year 2000 and you found a backup tape that has that file. Many restore programs will restore the mtime as well. When you run that program it will print an mtime from the end of the year 2000. But the ctime is today. So again, our backup program will see the file as needing to be backed up.

Suppose your restore program did not restore the mtime. You don't want your program to print today's date. Well no problem. mtime is under your control. You can set it to what ever you want. So just do:

Code:
$ touch -t 200012311800 employees.txt

This will set mtime back to the date you want and it sets ctime to now. You have complete control over mtime, but the system stays in control of ctime. So mtime is a little bit like the date on a letter while ctime is like the postmark on the envelope.


ctime-mtime-atime.pdf

Timestamp:

>> to display the current timestamp:
date '+%s' --> o/p: 1486357912 (current time Mon Feb 6 10:41:46 IST 2017)
>> to convert timestamp into date:
date -d "1970-01-01 1486357912 sec GMT" --> 1486357912 is the timestamp and 1970-01-01 is the base time.
>> To convert date into timestamp.
date -d "2016-02-06" '+%s' o/p:946684800

Resources: unix time stamp calculator : http://www.unixtimestamp.com/
++

Quotes

>> In shell there are 3 types of quotes avilable. single quotes (''), double quotes("") and back quotes(``).
>> the text embedded in single quotes('') will not be interepreted (except another quote.) the special characters of the shell like $, *, [, # have no meaning inside a single quote.
ex: $ echo 'The total is
nearly $750'
o/p: The total is
nearly $750

>> In double quotes, some special characters will work, for ex: $(for variable substitution), Back quotes (``) and also certain constructs like \$ or \" etc. when we use \ with some special character, it will not be interpreted.
ex: echo "$LOGNAME made \$1000 in `date +%B`"
o/p: systemout.log made $1000 in date November

>> Back quotes have nothing to do with the special characters. The text with in the back quotes treated as a command.


Double or Single brackets, parenthese

++

Logical AND – OR

>> command1 && command2 --> execute command2 only when command1 executes successfully, ie it gave the exit status of zero.
>> command3 || command4 --> execute command4 only when command3 does NOT executes , ie it gave the exit status of NON-ZERO.

>> ls file1 && cp file1 /tmp --> only when there is a file1 exits ie when the first command returns exit status of zero, run the 2nd command ie copy the file1 to tmp directory.

>> diff fileA fileB || echo "file A and fileB are different" --> if there is a diffrence ie, first command returns non-zero exit staus, ie, first command exit status is false, then run the 2nd commnand of echo.

>> ls fileA || exit --> if fileA does not exist, then exit.
++

$ Characters

```
$#    the number of arguments
$*    all arguments to the shell
$@    similar to $* except when quoted with "
$-    options supplied to the shell
$?    value returned by the last command executed
$$    the process-id of the shell
$!    the process-id of the last command started with &
$HOME the user's home directory; default argument to cd
$IFS  the list of characters which separate commands in arguments
$MAIL file which, when changed, triggers ``you have mail'' message
$PATH      list of directories to search for commands (: separated)
$PS1  primary shell prompt, default $
$PS2  secondary shell prompt (continued commands), default >
```

>> $? --> this indicates the exit status of the previous command.
>> Redirecting to "/dev/null" is a way to stopping the program from printing the output to screen.
>> !! --> On the command line, !! would be expanded to the last command executed. Bash will print the line for you:
>> input can be taken by keyboard(default), from another file (using <), or from the o/p of anohter program (using |)
>> output will be sent to screen (default) and it can be sent to a file (using >) or to a program (using |)
>> Each program produce both o/p and standard error.
>> ">" is used for the redirect to a file. "|" is used to redirect to a program.
>> ">>" is used for redirection + appending.
>> "<" standard input can be read from a file
>> "|" standard output can be piped to another program
>> "1>" can be used to redirect the Standard Output.
>> "2>" -> "Standard Error" is separate from "Standard output" 2> can be used to redirect the Standard Error to another file.
>> "&1" --> standard output and standard error can be routed to same file like the example below.
find . -name vivek.txt > results.log 2>&1 --> standard o/p to the results.log and stderr to the stdout, in this case to the same file results.log

>>Note that it is never possible to give a file a name that includes the / character

>> Anyform of o/p either standard o/p or standard error can be redirected to /dev/null, if it is not wanted at all.

++

Ulimit

>> ulimit -n 10240 (in ubuntu ulimit -n is not there.)

>> ulimit –a (to see the soft limits and hard limits)

Ref: http://stackoverflow.com/questions/24955883/what-is-the-max-opened-files-limitation-on-linux

++

Umask

What is umask?
>> umask is used to determine the permissions of the newly created files by the user. It is used to control the Default File permissions.
The default permission for root user is 002, and for normal user is 022.
For Directories base permission is 0777, for files base permission is 0666.That means if you set umask value to 002 then you subtract the umask from the base permissions:
For Directories :-->   777-002=775 is the permission
For files:-->   666-002=664 is the permission.
>> One can check the current value of umask is simply by typing "umask". and can be set the permissions with example like **"umask 0022"**

++

Uname

>> uname -a --> this gives all the details about the operating system.
>> "uname -n" gives the hostname in the unix machines.

++

Sudo & su

>> sudo and su are different ways to get the Root previleges.Both su and sudo are used to run commands with root permissions.

This is a key difference between su and sudo. **Su switches you to the root user account and requires the root account's password. Sudo runs a single command with root privileges – it doesn't switch to the root user** or require a separate root user password.

The su command switches to the super user – or root user – when you execute it with no additional options. You'll have to enter the root account's password. This isn't all the su command does, though – you can use it to switch to any user account. If you execute the **su bob** command, you'll be prompted to enter Bob's password and the shell will switch to Bob's user account. Once you're done running commands in the root shell, you should type **exit** to leave the root shell and go back to limited-privileges mode.

By default root account is locked under Ubuntu Linux. Therefore, you cannot login as root or use **'su -'** command to become a superuser. To run all administrative command use sudo command. sudo allows a permitted user to execute a single command as the superuser or another user. By default, Ubuntu remembers the password for fifteen minutes and won't ask for a password again until the fifteen minutes are up.

++

tar

>> **tar -cvf newarchive.tar Diroctory_of_files_tobe_archived** --> to create a archive file with the contents of diroctory_of_files_tobe_archived

>> **tar -xf newarchive.tar -C Directory_to_extract** --> note that Directory_to_extract should be existed before the command executes.

>> tar -tvf newarchive.tar --> it will list all files in the archive without extracting.

cpio:

cpio works like tar, but it can read input from the find command. it works better than tar.

>> find . -iname "*c" | cpio -o -H tar -F Cfile.tar   --> F:filename, H: format o:create tar file
>> cpio -i -F Cfile.tar --> extracts the files to the location where the Cfile.tar resides.
>> cpio -it -F Cfiles.tar --> it just lists the filenames in the Cfiles.tar archive.
>> (cd /your_target_dir && cpio -i path_to_archive/archive.tar) --> this will extract the tar file to the target directory.


tar examples.pdf

++

Cpu and Memory proc

>> cat /proc/cpuinfo --> this will give what type of processor/cpu you are using.
>> cat /proc/meminfo --> this will give stats on memory.
++

du-df

"du" means disk usage. and "df" means disk free.
>> df
>> df –h → human readable form of the free diskspace.
>> df –k → to get the logical volume information.
>> df -i → gives the percentage of inodes used in each disk volume.

>> du -hs *
>> du -hs /bin /lib /home --> to see the used space in multiple directories, "h" represent the human readable.
>> sudo du -m / | sort -k1n → to sort the folders based on size (MB)

++

FREE

>> free --> it display the free memory, swap and cache memory
>> free -h
>> free -mto --> total(t) memory and swap in MBs. "o" hides the buffer/cache memory.
++

Nslookup - Hostname

>> hostname -I --> gives all addresses for the host o/p: 192.168.1.20

>> hostname –i → gives ip address of the host o/p: 127.0.0.1

>> hostname –b → gives the boot name of the host o/p: ubuntu

>> nslookup <hostname> gives the ip address.

>>"**nslookup myip.opendns.com resolver1.opendns.com** " this command will gives you the external ip address if you are behind a NAT router.

++

NETSTAT

Ref: http://www.thegeekstuff.com/2010/03/netstat-command-examples/
----
a=all, l=listening, s=stats, p=pid, c=contineously, n=number, t=tcp, u=udp

>> netstat -at --> lists all ports on tcp protocol
>> netstat -au --> udp protocol.
>> netstat -l --> all listening sockets
>> netstat -lt --> listening tcp sockets.
>> netstat -p --> lists the pid of the program
>> netstat -an --> dispaly all the socks in numbered form. does not resolve into the names. ie, names will not be dispalyed.
>> netstat -c --> display the information contineously.
>> netstat -ap | grep 'ssh' --> find out ssh program is using which port.
>> netstat -an | grep ':80' --> find which program is using port 80.
+++\

top command:

Ref:http://alvinalexander.com/linux/unix-linux-top-command-cpu-memory

>> this command gives the dynamic real time view of the running system. It gives the processes or threads currently run by the Kernal and the system summery information. top command will give summery of tasks running, their pid, ppids, paths, how much cpu they are using, how much memory they are using etc.
>> q for quit, and h for help.

++

w:

This will give the **load avarage**. this command is present in almost all flavors of unix. It gives the quick view of the avarage load. below is the output of "w" command.

```
vivek@ubuntu:~/bin$ w
 19:56:42 up 12:27,  2 users,  load average: 0.22, 0.19, 0.21
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
vivek    :0       :0               Wed22   ?xdm?  38:22   2.64s cinnamon-
session --session cinnamon
vivek    pts/1    pc:S.0           Wed22    2.00s  4.36s  0.59s w
```

the o/p was taken at 19:56:42
the system is up for 12:27 hrs.
user count: 2
load avg for last minute: 0.22
load avg for last 5 minutes:0.19
load avg for last 15 minutes:0.21

If the load avarage exceeds 1, then the system is in overload.

++

Vmstat

This gives information about processes, memory, swap, IO, System, cpu information, This gives information about processes, memory, swap, IO, System, cpu information,

```
vivek@ubuntu:~/bin$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------
cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy
id wa st
 0  0    396  96440 128148 322040    0    0    12     3   46   89  4  1
95  0  0
```

++

IOSTAT

```
vivek@ubuntu:~/bin$ iostat
Linux 3.16.0-30-generic (ubuntu)        02/02/2017      _i686_  (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           4.45    0.01    0.63    0.31    0.00   94.60

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               0.99        11.46         3.28     510435     146148
```

++

Crontab

>> crontab -l --> to list out all the crontabs.
>> crontab -e <path/to/script.sh> → to create a crontab.
>> crontab –u <username> -e <path/to/script.sh>
>> 1st* 2nd* 3rd* 4th* 5th* command
1st* --> Minutes(0-59)
2nd* --> Hours(0-23)
3rd* --> Day of the Month (1-31)
4th* --> Month (1-12) 1=Jan,2=Feb etc.
5th* --> Day of the Week (0-6) 0=sun,1=mon etc.

>> */10 -- every 10th period. for ex: */10 * * * * cmd --> every 10 minutes run the command.
>> 10-18 --> run the command for every 10th,11th, 12th ...till 18th period.
Ex: * 10-18 * * * command --> run the command for every hour between 10am to 6pm.

>> * * * * 1-5 command --> run the command on mon,tues, wed, thur,fri ie every weekday.

>> @yearly (0 0 1 1 *), @daily (0 0 * * *), @hourly (0 * * * *), @reboot (Run at startup.)

++


crontab-reference.pdf

++

Ways of Executing a script

dot space dot slash method:
>> . ./myscript --> there is a space after the first dot. It tells that source("." - first dot) the myscript from the "."(2nd dot - current directory)

source method:
>> source ./myscrpt --> source the myscript form the current directory(".")

These above 2 methods allow the commands in the script to be executed in the same shell as the login shell. In the below 2 ways, the script will run in its own shell. so as long as the script execution complets, its variables are not availble to the login shell.

specifyingpecifying file name:
>> /home/vivek/myscript
>> ./myscript

A new shell will be loaded into the memory to run the script file.

Specifying the interpreter:
>> bash myscript
>> bash -x myscript allows to see which command is currently executing. it is helpful in debugging.
>> sh myscript

Exec
exec myscript

The current shell terminates, and spawns a new shell. When the script execution has been completed, the user will be logged off.
++

Variables – notes

>> name=vivek ---> there should NOT be any gap on either of side of =.
>> echo $name --> it prints vivek.
>> echo ${name} --> this also prints vivek.s
>> there are 2 ways we can retrieve the value stored in the variable. $variable, ${variable}. ${variable} is useful in certain printing tasks, for example, when we want to create a file name with variable_LOG, this can be done easily by
>> touch ${variable}_LOG

++

xargs vs exec:

>> Below is one major difference between xargs and exec:

find . -name "*.c" -exec ls -al {} \; executes the command ls -al on each individual file.

find . -name "*.c" | xargs ls -al constructs an argument list from the output of the find command and passes it to ls.

consider the below produced output of the find command:

a.c
b.c
c.c
the first command(exec) would execute

ls -l a.c
ls -l b.c
ls -l c.c

but the second (xargs) would execute

ls -l a.c b.c c.c

++

Tasks from the book " unix for oracle dba's":

These tasks list is improtant from the administrtaive point of view.
>> find the number of cpus.

13) Miscleneous scripts:
---

12)Disk Management:

>> Disks --> physical volumes --> physical partitions(pp) --> logical volumes --> maps to Unix Mount points. --> several logical volumes at Mount point is known as volume group.
>> Mount point is like a directory name or disk storage when allocating files.
a) List logical volumes, mount points
df –k

11) File Management:

a) List recently touched files. **ls -alt | head 10** --> t option makes sort by the recently touched. Touched means, the file was read by a process. It need not necessarily changed.
b) List recently changed files. **ls -alc | tail** --> c option makes it to sort based on change of the date and time.
c) Delete unchanged files.
find . -ctime +5 -exec rm {} \; --> -ctime +5 means, change dates more than 5 days in the past.
d) display files of 512 KB blocks. --> **du -s** --> s means summery.
e)locate files that contain certain strings.
find . -type f -print0 | xargs -o grep -i "string" {} \;
f) find recently created files.
find . -mtime -1 ---> -1 means, age less than one day old. +1 means, age more than one day old.
g) find large files on a server
find . -size +10000 --> +10000 bytes means bigger than 10KB
h) Delete files in bulk.
find . -mtime +7 | xargs rm {} \;
i) Allocate an empty file.
touch testfile
j)Change default file permissions.
umask command to be used.
k)change file ownership.
chown -R vivek vivek * this will change the ownenship and group membership to vivek vivek for all the files.

++

TEST

test value --> returns true if the value is not zero.
test value1 = value2 --> test if value1 and value2 are same. if they are same the stutus (echo $?)will be zero.
test value1 != value2 --> tests if value1 and value2 are not same. if they are NOT same, then staus will be zero.
test value1 -gt value2
test value1 -ge value2
test value1 -lt value2
test value1 -le value2
test -z value --> returns true if the value is empty (zero-length)
(x=20; test -z "$x"; echo $? --> returns false since x is not empty.)

test -f filname → test if it is a file and existed.

test -d filename → test if it is a directory.
test -s filename → test if it is a file and NOT empty.
test -z filename --> test if file existed and empty.
test -r/w/x filename
test expression → test if the expression is true.
test ! expression --> test ! -x config.sh --> test wether the config.sh is not executable.
test expression1 -a expression2 --> test if both expressions are true.
test expression1 -o expression2 --> test if either one of the expression is true.

++

TRUE & FALSE

>> **0** represents there is no error or it is **true**
>> any thing other than zero represents there is an error.
>> **$?** gives the **exit status** of the last command.
>> there are 2 programs called true, false which does nothing except sends 0 and 1 exit codes to the calling programs. These programs are useful in running infinite loops.
++

IF

if [-f ./vivek.tar] --> this is checking if there is a file called vivek.tar exists and is it a regular file.

++

while:

```
 ---------
 while condition
 do
 commands
 done
---------------------------------------
```

while has another way of operating. it can read output of each line and process the line each one at a time.

```
 who | while read user terminal time
> do
> echo the $user has been on $terminal since $time
> done
the vivek has been on :0 since 2017-02-01 22:42 (:0)
the vivek has been on pts/1 since 2017-02-01 22:51 (pc:S.0)
```

++

CASE

```
case $var1 in
  $value1)
             code for case1
             ;;
  $value2)
             code for case2
             ;;

  $value3|value4|vlaue5)
             code for case3 or value4 or value 5
             ;;

  [vV]al*)
           code for any val* or Val*
           ;;

   *)
           code for default case
           ;;
esac
```

++

Ctrl-m:

When copying files from windows to unix, the cr-lf (carriage return-line feed) of the windows will be presented as ^M.(press control and type simultaneously v and m) in unix. It will make the scripts non-executable. To overcome this, there is a sed command to remove these.

>> sed -e "s/^M//" file_whihc_has_ctrl_M > newfilename
>> mv newfilenamme original filename

You can also do it in vi: % **vi filename**
Inside vi [in ESC mode] type **:%s/^M//g**

>> To type ^M in this command press control and press v and m simultaneously.

++

SCP –PSCP:

Ref:http://www.hypexr.org/linux_scp_help.php
https://www.bitvise.com/configuring-ssh-server-for-sftp

>> install the scp server on the host windows machines. btvise ssh server will be useful in this purpose. you can create virtual users or os level users can be used. in the virtual file system layout, we can share/mount various folders.

Copy the file "foobar.txt" from a remote host to the local host

```
$ scp your_username@remotehost.edu:foobar.txt /some/local/directory
```

Copy the file "foobar.txt" from the local host to a remote host

```
$ scp foobar.txt your_username@remotehost.edu:/some/remote/directory
```

Copy the directory "foo" from the local host to a remote host's directory "bar"

```
$ scp -r foo your_username@remotehost.edu:/some/remote/directory/bar
```

Copy the file "foobar.txt" from remote host "rh1.edu" to remote host "rh2.edu"

```
$ scp your_username@rh1.edu:/some/remote/directory/foobar.txt \
your_username@rh2.edu:/some/remote/directory/
```

Copying the files "foo.txt" and "bar.txt" from the local host to your home directory on the remote host

```
$ scp foo.txt bar.txt your_username@remotehost.edu:~
```

Copy the file "foobar.txt" from the local host to a remote host using port 2264

```
$ scp -P 2264 foobar.txt
your_username@remotehost.edu:/some/remote/directory
```

Copy multiple files from the remote host to your current directory on the local host

```
$ scp your_username@remotehost.edu:/some/remote/directory/\{a,b,c\} .
$ scp your_username@remotehost.edu:~/\{foo.txt,bar.txt\} .
```

scp Performance

By default scp uses the Triple-DES cipher to encrypt the data being sent. Using the Blowfish cipher has been shown to increase speed. This can be done by using option -c blowfish in the command line.

```
$ scp -c blowfish some_file your_username@remotehost.edu:~
```

It is often suggested that the -C option for compression should also be used to increase speed. The effect of compression, however, will only significantly increase speed if your connection is very slow. Otherwise it may just be adding extra burden to the CPU. An example of using blowfish and compression:

```
$ scp -c blowfish -C local_file your_username@remotehost.edu:~
```

>> to copy a file from local windows server to remote linux machine. give this example commmand:
pscp "C:\Users\Vivek\Desktop\New folder\publickkey"
vivek@192.168.1.2:/home/vivek

C:\Users\Vivek>pscp "C:\Users\Vivek\Desktop\New folder\publickkey"
vivek@192.168.1.2:/home/vivek
vivek@192.168.1.2's password:
publickkey | 0 kB | 0.5 kB/s | ETA: 00:00:00 | 100%

>> To copy from a remote linux server to local windows machine, give the below commmand.

C:\Users\Vivek>pscp vivek@192.168.1.2:/home/vivek/Desktop/publickkey
"C:\Users\Vivek\Desktop\New folder\publickkey"
vivek@192.168.1.2's password:
publickkey | 0 kB | 0.5 kB/s | ETA: 00:00:00 | 100%

>> to copy a directory, use "r" flag. r=recursive.
$ scp -r foo your_username@remotehost.edu:/some/remote/directory/bar --> it wil copy the local directory called "foo" to the remote directory "bar".

>> copy multiple files
$ scp foo.txt bar.txt your_username@remotehost.edu:~ --> it will copy foo.txt, and bar.txt
>> $ scp your_username@remotehost.edu:~/\{foo.txt,bar.txt\} .

scp reference.pdf

++

Vi Editor Basics:

>> Noral mode - ESC key. Always press ESC key to come to normal mode.

:q! -- Exit with out saving anything.
SHIFT + zz -- Save and exit
:w! -- Save and does NOT Exit, allows to edit still, saves till that point.

Insert mode - press "i"(at the current cursor location) "I" will move the cursor to starting of line.
Append Mode - press "a" (at the current cursor location)   "A" will go the end of the line.

>>

x - it will delete a single character. hit x repetedly to delete characters.
dd - removes a line.

>> r -- replace mode. But only single character it will allow you replace. "R" will allow in the true Replace mode. Remeber to press ESC to come to Normal mode once you have done the job.
>> "J" joins the next line with the previous line.

>> Search in the fil: "/" to search forward and "?" to search in the backward direction. "n" for next result and "N" for previous result.

yy - to copy a line and "p" to paste it.

v - to select a block of text. and "yy" to copy it (after pressing v, and text selection, do not press esc. press yy immediately.) and then "p" to paste it.

>> **:!filename** --> this is to run the same script while inside the vi editor without exiting. this is useful while doing a quicktesting without exiting the vi editor.
>> **:!** This allows a single command to be run inside the vi.
>> : pwd or :! Ls –l
>> **:!!** --> this is to repeat the last command in the vi.

++

PUTTY – Settings:

>>Start the Putty. you will login to the unix server with port 22(SSH).

>>enter the ip address or hostname, and hit open.

>> X11 Forwarding: In the putty, Open the "Connection" tab --> X11 --> check the box for "Enable X11 forwarding", x-display location is "localhost:0". Run the xming server in your windows machine from where the putty session has been started. This will enables the X11 forwarding. and open gedit for the notepad++ type of editor.

>> install open-ssh server in ubuntu. /etc/ssh/sshd-conf is the server config file and ssh-conf is the client config file. take the back up of the sshd-conf file. WHAT CHANGES THAT I DID TO THE CONFIG FILE? (one change: remove the commment for the authorized_keys file location.)

>> sudo service ssh restart --> to restart the ssh server.

>> C:\OpenSSH>ssh-keygen.exe -t rsa --> this command also generates public and private keys.

>> sskkeygen will generate the public and private keys. i have not given any password. $home/.ssh/authorized_keys file should be having the public key. .ssh have chmod of 700, and authorized_keys file should have chmod 600. (Note that authorized_keys is a file not a directory).

>> in the putty client, connection -->ssh--> point to the private key lociton and save the session settings. in the data field, give a username to login to.

>> use the screen program. install the screen program in the ubuntu. open the putty, type screen to start the screen program. while in the middle, type "Alt+A, and d". it will detach the screen. then you can log out also from the ssh session from putty. This screen will get detach itself. When you want to re-attach, login to putty, and type "screen -r"

++

Ssh server- bitvise – public keys :

>> in the unix machine cd to ~/.ssh
>> ssh-keygen -t rsa
>> asks for passphrase, given basara234
>> id_rsa and id_rsa.pub will be genereated in /home/vivek/.ssh
>> id_rsa holds the private key protected by the passphrase of basara234 and id_rsa.pub has the publick
key which must be copied to the ssh server machine.
>> in the bitvise ssh server easy settings --> virtual accounts --> select the virtual user --edit--> public keys --> import this id_rsa.pub
>> for setting up the public keys in unix ssh server with putty client, follow the chrome browser bookmarked article in ws--> putty
>> from the unix command prompt to ssh to bitvise ssh server on the host windows machine, "ssh shiva@pc" this will ask for the passphrase of the private key id_rsa, which is basara234.

++

Screen

Screen program allows us to have multiple screens and in each screen, multiple windows.
Ref: http://fosshelp.blogspot.in/2014/02/linux-screen-commands-for-developers.html
---
Create new session:
---
screen --> creates an unNamed screen session.
screen -S Name_of_the_session --> creates a named screen session.

Attach & Detach & reattach
---
ctrl a,d --> detach the screen

screen -r <session name> --> to reattch the screen
screen -x <session name> --> to reattch the screen for multiple user.

Terminatation:
---
ctrl a, \ --> terminates all sessions.
ctrl a, shift k --> terminates a window.

Lock& Scroll:
---
ctrl a, x --> to lock a screen .
ctrl a, esc --> scroll in the screen. Hit Enter to come back to the commmand prompt.

Widows in the screen:
---
ctrl a , c --> creates a new window.
ctrl a, shift a --> to rename a window.
ctrl a, shift ' -->to list all the windows in the curren screen.

Split windows:
---
ctrl a, shift s --> split the current window horizontally.
ctrl a, shift \ --> split the current window vertically.

Window Movement:
---
ctrl a, ctrl a --> to toggle between windows.
ctrl a, Tab --> This will move to the other windows in the splitted screen.
ctrl a, 0-9 --> to move to the numeberd window.
ctrl a, ' --> to go the nth window.
ctrl a,p--> to move to the previous window.
ctrl a, n --> to move to the next window.

ctrl a, shift c --> will clear the screen.
ctrl a , shtift x --> clears the current window in splitted screen.
ctrl a, shift q --> clears all the windows except the current one.

ctrl a, shfit h -->logs the current window.

Screen – Reference

Reference:http://fosshelp.blogspot.in/2014/02/linux-screen-commands-for-developers.html
1)
List all screen sessions of current user
#screen -ls

2)
To see all screen sessions on a specific machine
#ls -laR /var/run/screen/

3)
To see all commands or parameters on screen.
Ctrl + a, Then Press ?

4)
Open a new screen session
#screen
5)
Detach/Exit from a screen session
Ctrl + a, Then Press d

6)
Kill a screen session
Ctrl + a, Then Press Shift + k

7)
Reattach to a screen session
#screen -r session_id_or_name

8)
How to create multiple screen window/tab in a screen session
Ctrl + a, Then Press c

9)
How to move to next screen window/tab in a screen session
Ctrl + a, Then Press n

10)
How to move to previous screen window/tab in a screen session
Ctrl + a, Then Press p

11)
How to goto n'th screen window/tab
Ctrl + a, Then Press '

12)
How to list name of all screen windows/tabs and select from there
Ctrl + a, Then Press Shift + '

13)
How to switch to screen window/tab 0 - 9
Ctrl + a, Then Press 0 -9

14)
How to Toggle to the window/tab displayed previously
Ctrl + a, Then Ctrl + a again

15)
How to change the name of screen window/tab
Ctrl + a, Then Press Shift + a

16)
How to clear a screen window/tab
Ctrl + a, Then Press Shift + c

17)
How to Kill all windows and terminate screen
Ctrl + a, Then Press \
screen -ls

18)

How to lock a screen session
Ctrl + a, Then Press x

19)
How to log a screen session
You will find screenlog.0 file in your home directory.
Ctrl + a, Then Press Shift + h
or
#screen -L (Capital L)

20)
reattach to a specific window/tab in a screen session
#screen -r session_id_or_name -p tab_num_or_name

21)
If a session is running, then reattach. If necessary detach and logout remotely first. If it was not running create it and notify the user.
#screen -D -R

22)
How to create new screen session and specify a meaningful name for the session.
#screen -S session_name

23)
Howto Attach to a not detached screen session.
#screen -x

24)
How to split screen
a)
Split the Window
Horizontally
Ctrl + a, Then Press Shift + s
or
Vertically
Ctrl + a, Then Press Shift + \

b)
Switch between spilted windows
Ctrl + a, Then Press Tab
or
Ctrl + a, Then Type :focus
* Here :focus is a command

c)
In the spited window use following command to open existing session
Ctrl + a, Then Press 0-9
or
Ctrl + a, Then Press n or p
or
Ctrl + a, Then Press Shift + '
or
Ctrl + a, Then Presss c

d)
Resize a splitted window/region
Ctrl + a, Then Type :resize 25

```
* Here :resize is a command

e)
Remove current splitted window/region
Ctrl + a, Then Type :remove
* Here :remove is a command
or
Ctrl + a, Then Press Shift + x

f)
Remove all spiltted windows/regions except the current one.
Ctrl + a, Then Type :only
* Here :only is a command
or
Ctrl + a, Then Press Shift +q

g)
Change color of split bar (border)
http://michael-prokop.at/computer/config/.screenrc
#vim ~/.screenrc
sorendition 10 74

h)
Close the screen and all regions
Ctrl + a, Then Press \

25)
How to rename an existing session
screen -ls
screen -x old_session_name
Ctrl + a, Then Type :sessionname new_session_name
*Here :sessionname is a command

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Services

```
>> to check the status of service running or not, use the below command.
+ means service is running, - means not running, ? means status is not
determined.

>> sudo service --status-all
```