**Youtube: Shell Scripting Tutorials by The Bad Tutorials**

**PART – I**

**COMMANDS**

**SHELL SCRIPTING**

-**Linux** is NOT Unix. Unix was developed in 1969 by AT&T while Linux was developed by a single man in 1991 (Linus Torvalds).

-Linux is **free**

**-**First, Available free of cost

-Second, Freedom to manipulate the source dode.

-Linux is **open-source**

**-**Linux is developed under GNU Public Licence (Copyleft)

**-**Under this Licence:Source code is available to all and they can be modifies, developed and Sell **BUT** you have make your code open source as well.

-Linux is **Unix Like**

**-**Like Unix, Linux is written in C.

-Like Unix, Linux is also the Multi-user/Multitasking/32 or 64 bit Network OS.

-Like Unix, Linux is rich in Development/Programming environment.

-Like Unix, Linux runs on different hardware platform eg: Intel x86 Platform

-Linux is **Network OS**

- Linux Distributions available: Ubuntu, Mint, fedora, ANDROID

**INTRODUCTION**

**Process**

Programming language: Learn alphabhabets --> Make Words ---> Make Instructions --> Make prog

Shell Scripting: Learn Commands ---> Use commands to build programs

**Terminology**

**A) SHELL:**   
  
- Interface between us and kernel  
- We write programs for shell (Shell Script), Shell inturn translates into something Kernel can understand and pass it to Kernel (who is responsible for allocating resources).

- Its a COMMAND LINE INTERPRETTER.

**B) KERNEL:**

- Its a program that manages system's resources or hardware.

- It takes instructions FROM shell and TRANSLATE it into something machine can understand and get the job done.

**C) TERMINAL**

**-** Shell is accessed via an Application called Terminal.

- It has a CLI (Command line Interface)

-In terminal:  
<Username>@<name of machine> :~$

Here, dollar represents shell.

**D) SHELL SCRIPT**- Its is a computer program designed to be run by the Shell, a command line interpretter  
  
**E)**  **Scripting Language  
  
-** Its a programming language that supports **scripts**, programs written for a special runtime environment that can INTERPRET(rather than compile) and automate the execution of tasks that could alternatively be executed one-by-one by a human operator.

1. who am i ---> To know the time at which I (user ) logged in.

2. pwd (present working directory) ----> Tells the dir in which shell currently is.

3. cal

- cal ---> current cal

- cal <month> <year> ----> eg: cal 3 2004 OR cal mar 2004

4. date

-It can be customized to change the format of viewing the date

5. clear

6. **uname -a** Linux skand-Lenovo-B480 3.13.0-63-generic #103-Ubuntu SMP Fri Aug 14 21:42:59 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux  
  
-Displays info about version of kernel, system info, processor info, version of OS etc.

6. **touch** <file 1> <file 2>.......n (To **CREATE** BLANK Files)

-It creates empty blank text files/file in pwd (i.e type text/plain.)

-eg: touch blank1 blank2 blank3

7. <command> --help

8. **mkdir** <folderName> ------> eg: mkdir test : creates test folder in pwd (To **CREATE** dir)

- mkdir <Path> ----> eg: mkdir Documents/Floder: creates folder in home/Documents

9. cd <Path> -----> change directory

10. **cat (**To **CREATE/MERGE** files WITH CONTENT)

(A) cat > <Textfile to be searched/created>

eg: cat > newFile : shell searches for file named newFile and opens/creates it AND you're prompted to type text in the file. SAVE it by pressing **Ctrl+D**

(B) cat < <TextFile whose content is to be viewed> -----> eg cat < newFile

(C) cat <File1> <File2> > <NameOf **Merged** File> ------> It merges two files.

Eg: cat > file1

This is file 1.

cat > file2

This is file 2

cat file1 file2 > mergedFile

cat < mergedFile

This is file 1.

This is file 2.

11. **mv** <NameOfFile Or Directory to be renamed> <New name> (To **RENAME** a file/dir)

eg: mv File FilNewName ---> renames File with FilNewName

12. **rm** (To **DELETE** a file/dir)

(A) rm <fileToBeDeleted> (Deletes ONLY files)

(B) **rm -r** <DirToBeDeleted> (Deletes Directory) OR **rmdir** <DirToBeDeleted>

13. **cp** <file> <PATH **> (** To **COPY** file from one loc to another loc)

eg: cp old Music/old\_copy ---> copies “old” file from pwd and pastes it in Music dir and give it a name “old\_copy”

14. **ln (LINKS** for FILES & DIR)

(a) touch old (Only valid for FILES)

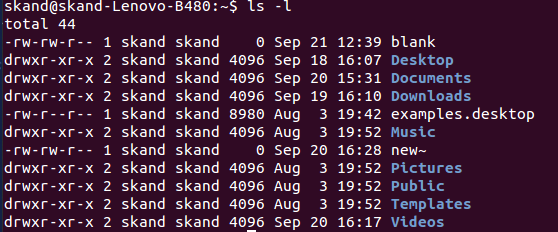
ln old link // Creates a file “link” which is **HARD-LINK** for the file “old” i.e Its a replica of the file “old” any changes in “old” will be reflected in link” i.e if “old” is deleted “link” still exists in same condition as a BACKUP.

(b) touch old (Valid for FILES & DIR)

ln -s old soft\_link //creates a file “soft\_link” which is **SOFT-LINK** to the file “old” i.e its not a replica but just a pointer to “old” i.e if “old” is deleted “soft\_link” is useless.

**FILE PERMISSIONS**1) Read Permission (You can only read the file) -----> Numeric value: 4  
2) Write Permission (You can make changes in the file) -----> Numeric value: 2  
3) Execute Permission (You can execute the file as a program) -----> Numeric value: 1  
  
**Note:**A file at best can have total numeric value of 7 (4+2+1 when it has all 3 permissions)  
  
Three groups of user can access this file:  
  
1. Owner (Default Permissions: Read(4) + Write(2) = Numeric Value: 6)  
2. Group to which owner belong (Default Permission: Read(4) + Write(2) = Numeric Value: 6)  
3. Others on the OS (Default Permissions: Read(4) = Numeric Value: 4)  
  
**Note:** Default Numeric Value for a file -----> **664**  
  
**Note:**System variable **umask  
  
$** umask  
--> 0002  
  
// First 0 represents octal number  
// The last 3 nos help decide the default permissions for a file: 0002  
// METHOD:   
  
Default Permissions for a file = 666 – umask  
 = 666 – 002  
 = **664**

15. **ls (**To **DISPLAY/list**  all files & directories) **-** It list all FILES and FOLDERS in the directory.  
  
(A) ls ----> displays all files & folders  
(B) ls <PATH> ----> eg: ls Documents ---> It displays all files & folders in Documents dir.  
  
**Note:**(C) **ls -l** -----> -l is OPTION for lonlisting



**t*otal 44 :***

-Its the total no of BLOCKS occupied by all the files & folders in this dir.

-**Block**  is a unit of m/m that is used by Linux (1 Block= 1024 Bytes maybe)

***First Column:***

**-** A hyphen(-) in first column indicates its a normal file

- A “d” in first column indicates itss a directory

***Columns 2-10:***

- These 9 columns are in group of 3 indicating permissions for owner, group and others repectively.

-eg: d **rwx** r-x **r-x**

“d” indicates its a directory.

“rwx” indicates read, write, execute permissions for owner.

“r-x” indicates read & execute permissions for group.

“r-x” indicates read & write permission for others.

**Note:**

**- Hidden Files** can be viewed by pressing Ctrl+H in GUI.

- To hide a file save its name preceded by dot(.) and refresh. Eg: .new is a hidden file and can be viewed after pressing Ctrl+H

**(D) ls -a** ----> a means all (i.e hidden files will be viewed as well)

**16. file \*** ---> \* means all (To **DISPLAY** all files & dir **WITH** *type* of its content)

blank: empty

Desktop: directory

Documents: directory

Downloads: directory

examples.desktop: UTF-8 Unicode text

Music: directory

new~: empty

Pictures: directory

Public: directory

Templates: directory

testBlank: empty

Videos: directory

-It displ

**CHANGE FILE PERMISSIONS**

16. **chmod**

- chmod <numerical value> <file name>

eg: chmod 777 test ---> It will provide all 3 permissions (r,w,x) to all (owner, group and others)

**Note:**

Q. Significance of execute permission for a director?  
A. You can enter it and make changes, add new files and folder(If you have write permission).

If you have r,w but not x permission for a dir THEN you cant do anything with the dir.

**COUNT WORDS,LINES & CHARACTERS**

17. **wc <filename>**

**O/P : <#lines> <#words> <#characters> <Name of file>**

(a) wc -l <filename> // To print just the Lines

(b) wc -w <filename>

(C) wc -c <filename>

**SORT**  (Sorts the **SENTENCES** in alphabetical order)

a) **sort** //It then waits for you to type unless you press Ctrl+D

I/P:

I am dead

I am alive

hero

Bombing of me

Aviator dies

O/P:

Aviator

Bombing of me

I am alive

I am dead

hero

b) **sort <filename>**

**-**It sorts the sentences of the given file in above mentioned way.

**FILTER OUT CONTENT OF A FILE**

19. **cut**

**Example:**

~$ cat > playersFile

Name-Game-Age

Sachin Tendulkar-Cricket-40

Sanath jaisurya-Cricket-42

Tiger Woods-Golf-35

James Bond-Actor-55

Michael Corleone-Mafia Boss-75~$ cut -d"-" -f 1,3 playersFile

Name-Age

Sachin Tendulkar-40

Sanath jaisurya-42

Tiger Woods-35

James Bond-55

Michael Corleone-75

**Note:**

**cut -d”-” -f 1,3 playersFile**

**-d ---->** is used to specify delimiter. By default its TAB. For CSV files it will be comma.

**-f ----->** is used to specify what columns you want to be extracted

**playersFile** ----> Name of the file

**COPYING & CONVERSION**

20. **dd**

a) **dd if=<file1> of=<newFile> conv=ucase (UPPERCASE** Conversion**)**

//This converts Input file(**if**) to UPPERCASE and save the result in OutputFile(**of**)

a) **dd if=<file1> of=<newFile> conv=ebcdic (**change **CHARACTER ENCODING of a file)**

//This converts Input file(**if**) to european format and save the result in OutputFile(**of**)

21. **man <command-name> (**to Get **Help**)

//to get documentation page of a command. Eg: man cat

22. banner (to View Text in **fancy** Form)

a) banner <Text>

-Displays text in fancy form BUT it interprets space as new line.

b) banner “<Text>”

-Displays text in fancy form BUT only 10 characters and ignore the rest.

**COMPRESSING/UNCOMPRESSING FILE SIZE**

23. **compress -v <filename> -------**> -b is to view how much compression is achieved  
 -The file is compressed and it is renamed with **filename.z**

**Note:**

-You can't view the COMPRESSED file using cat filename.z anymore because it will display unprintable characters.

-To **VIEW CONTENTS of COMPRESSED FILE** you have to use **zcat** command

24. **zcat <compressedFileName>**

**25. uncompress <CompressedFileName>**

-It decompresses the compressed-file AND original name is retained.

-eg: zcat bulky.Z

26. **echo <text>--->** to display text on screen.

**Youtube: Shell Scripting Tutorials by The Bad Tutorials**

**PART – II**

**SHELL PROGRAMMING**

**BASIC STUFF**

**Convention:** Extension should be **.sh** for Shell Script

**Comments:** # is used as comment.

**Display text on screen:** Use “echo” command.

**Executing a script file :** $ **sh** <scriptFileNameWithExtension>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SCRIPTS**

1) Script 1: **ss1.sh**

**#**This is a script file

**echo** Hello World

**echo** The END

**Note:** echo command puts \n at the end by default.

**Note:**   
  
**SHELL VARIABLES**

case sensitive, start with number or underscore but caan have number in middle, containers for values

2) Script2 : ss2.h

**#**Script

**echo** Enter your name:

**read** my\_name **#**created a variable my\_name + read command takes input from User

**echo** Hello **$my\_name**, Get out **#** $ is used to access variable