1. **cat**

* #cat>file\_name (To create the file and write the content)

Enter the content…..

………………………………..

Press Ctrl + D to save the content and exit.

* #cat filename (To read the data in the file)
* #cat file1 file2 (To read the multiple files)
* #cat -n filename (To view the content with line numbers)
* #cat>>file1 (To append the data to an existing file(file1) )
* #tac filename (To display the content in reverse order)
* #cat -E filename (It displays “$” symbol at the end of each line)
* #cat -- -filename (To read the data in the dashed files)

Ex: cat -- -file3

* #cat filename|more ( If the file has lot of content and can’t fit in the terminal(| pipe symbol))
* #cat file1 file2>file3 (To copy content present in the file1 and file2 into the file3)
* #cat \*.txt (Displays all the txt files data in a directory)
* #cat -v filename (Displays the non-printing characters)

1. **ls**

* #ls (Displays all the files and directories)
* #ls -l (Displays all the files and directories in a long format i.e., detailed info about files and directories )
* #ls -a (Displays all files and directories including hidden)
* #ls -t (Sort the files and directories by their last modification time, displays last recent modification ones first)
* #ls -S (Sort the files and directories by their sizes, displays the largest ones first)
* #ls -R ( Displays the content in sub-directories too)
* #ls -i (Displays the index (inode) of each file and directory)
* #ls -r (Reverse the default order of listing)
* #ls -g (Displays the group ownership of files and directories instead of the owner)
* #ls -h (Prints the file sizes in human readable format)

1. **Pwd**

* #pwd (Present working directory)
* #pwd -L (Resolves the symbolic links and prints the path)
* #pwd -P (Displays the actual path without resolving the symbolic links)

1. **ll**

* #ll (long listing, it includes all hidden files and directories)

1. **mkdir**

* #mkdir directory\_name (Creates the directory in the filesystem)
* #mkdir dir1 dir2 dir3 (We can create multiple directories at a time )
* #mkdir -v directory\_name (It displays a message for every directory created)
* #mkdir -p existing\_dir/new\_dir (Creates the new directory within the existed directory , if existing directory is there then no error specified)

Ex: #mkdir -p demo1/demo2 Here demo1 is existing directory and demo2 is new directory which will be created in the demo1

* #mkdir -m a=rwx dir\_name (Used to set the file modes i.e., permissions)

1. **cd**

* #cd directory\_name (To efficiently move from one dir to another dir)
* #cd dir1/dir2/dir3 (To move inside a directory)
* #cd ~ (It change the directory to the home from any location in linux)
* #cd (We can pass the cd with no arguments which will lands us into the home directory)
* #cd .. (To move to parent or one level up from the current directory)
* #cd “My Directory” (This is used to navigate to a directory containing white space in the directory name)
* #cd My\ Directory (We can also use “\” symbol in between the white spaces , if we don’t want to use the single or double quotes)

1. **rmdir**

* #rmdir dir\_name ( It removes the directory from the filesystem, the directories must be empty)
* #rmdir dir1 dir2 dir3 (Removes multiple directories at a time, the directories must be empty)
* #rmdir -p dir1/sub\_dir1 (It removes the directory along with subdirectory)
* #rmdir -v dir\_name ( If we want to display the message after removing the directory)
* #rmdir krishna\* (Removing the directories with the same expressions by using “\*” )

Ex : root@ip-172-31-12-70:/home/ubuntu# mkdir krishna1 krishna2

root@ip-172-31-12-70:/home/ubuntu# ls

-bca.txt 1file a abc ak.txt b b~ crish demo file2 google k.txt krishna1 mer

-file3 'My Playlist' a.txt abc.txt ashoo b.txt c def file1 folder1 j.txt krishna krishna2 numbers

root@ip-172-31-12-70:/home/ubuntu# rmdir krishna\*

root@ip-172-31-12-70:/home/ubuntu# ls

-bca.txt 1file a abc ak.txt b b~ crish demo file2 google k.txt numbers

-file3 'My Playlist' a.txt abc.txt ashoo b.txt c def file1 folder1 j.txt mer

* #rmdir --ignore-fail-on-non-empty dir1 (To ignore the occurrences due to the non-empty directories

1. **cp**

* #cp source\_file destination\_file (It copies the content from the source file to the destination file, if the destination file does not exist then it creates new one , if exists overwrite it)
* #cp file1 file2 directory1 (It copies the file1 and file2 into the directory1, here the directory must be exist in the file system)
* #cp dir1 dir2 (It copies all the files from dir1 to the dir2 , if dir2 doesn’t exists it will create the new one)
* #cp -i source\_file dest\_file (It warns the user before overwriting the destination file , if we press “y” it will proceed)
* #cp -b source\_file dest\_file ( It creates the backup of the destination file in the same folder with different name)
* #cp -f src\_file dest\_file (If the system unable to open the destination file for writing operation)
* #cp -r (or) -R (It copies the directory structure)
* #cp \*.txt dir\_name (It copies all the text file into the destination directory)

1. **mv**

* #mv src\_file dest\_file (Renaming a file)
* #mv src\_file dest\_path (Moving a file into a destination path)
* #mv src\_dir\_name dest\_dir\_name (Renaming a directory)
* #mv -i src\_file/dir dest\_file/dir (Asks about overwrite , if the file name exists)
* #mv -f src\_file/dir dest\_file/dir (Overwrites the destination file forcefully and deletes the source file, if it is protected)
* #mv -b src\_file/dir dest\_file/dir (Take a backup of the existing file that will be overwritten, if file name already exists)

1. **rm**

* #rm file1 ( Removing a file)
* #rm file1 file2 file3 (Removing multiple files at a time)
* #rm -i file1 (Asks the user before removing a file)
* #rm -f file1 (Remove forcefully if it is write protected)
* #rm -r (\*) dir\_name (Removes even if it is directory, \* represents all )
* #rm -- -file1 (Remove dashed files in the filesystem)

1. **uname (To display system information)**  Displays OS version, OS release, hardware name and processor type.

* #uname (Displays info about the system)
* #uname -a (Prints all the system info in detail)
* #uname -s (Prints the kernel name)
* #uname -n (Prints the host name of the network)
* #uname -r (Prints the kernel release date)
* #uname -m (Prints the machine h/w name / type of processor)
* #uname -o (Prints name of Operating System)

1. **locate**

* #locate file1 (Search a file with a specific name)
* #locate “\*.txt” -n 20 (Limit the search queries to a specific number)
* #locate -c [.txt]\* (Displays the number of matching entries)
* #locate -i \*File1.txt\* (Ignore case sensitive locate outputs)

1. **touch**

* #touch file1 file2 file3 (Create multiple files at a time)
* #touch -a file1 (Changes the access time)
* #touch -c file1 (To check whether the file is created or not, if not created then do not create it)
* #touch -c -d ’22 Aug’ file1 (It is used to update access and modification time)
* #touch -m file1 (Used to change modification time only)
* #touch -d “22 Aug 2023” file1 (To change only modification date)
* #touch -r file1 file2 (The file2 is updated with the time stamp of file1)
* #touch -t YYMMDDHHMM (Create a file using specified time)
* #stat file\_name (To view the complete details about the file)

1. **ln**

* #ln filename linkname (Creating hard links)

Ex: #ln crish krish

* #ln -s filename linkname (Creating soft links)
* #ln filename dir\_name (Moving a link into directory)
* #ln -v filename dir\_name (Displays which file is linked to which file)

1. **clear**

* #clear (Used to clear the terminal screen)
* Press “ Ctrl + L “ to clear the screen

1. **ps**

* #ps (Used to check the active process in the terminal)

1. **grep : Global search for the regular expression**

* #grep “string” filename (Search for a string in the file)
* #grep -e “string1” -e “string2” filename (To find the mutliple keywords in a file)
* #grep -i “vamshi” file1 (Search for a string case insensitive)
* #grep -c “vamshi” file1 (Displays the count of no.of matches)
* #grep -w “vamshi” file1 (Doesn’t matches the substring words)

Ex: root@ip-172-31-213-198:~# cat f1

Hii

Welcome to the linux tutorials

Bye frnds

Hii

linux frnds

root@ip-172-31-213-198:~# grep -w "nux" f1

* #grep -o “vamshi” file1 (Displays only the matched patterns)
* #grep -n “vamshi” file1 (Shows line numbers while displaying the output)
* #grep -v “vamshi” file1 (Displays the lines that are not matched with the specified strings)
* #grep “^hi” file1 (Matching the lines that starts with the string)
* #grep “na$” file1 (Matching the line that end with the string)
* To Export the grep output to a file (#grep “string” filename >> output\_filename)
* #grep -A [no.of lines(n)] [“string”] [file\_name] (Prints the searched line and n lines after the result)

Ex: grep -A 2 “Vamshi” demo.txt

* #grep -B [no.of lines] [“string”] [file\_name] (Prints the searched line and n lines before the result)
* #grep -C [no.of lines] [“string”] [file\_name] (Prints the searched line and n lines before and after the result)

1. **echo**

* #echo “krishna” (Prints the string in the terminal)
* #echo -e “Vamshi \bKrishna” (Removes all the spaces in between the text)
* #echo -e “Vamshi \nKrishna” (Creates a new line)
* #echo -e “Vamshi \tKrishna” (Creates a horizontal tab space)
* #echo -e “Vamshi \vKrishna” (Creates a vertical tab space)
* #echo -e “Vamshi \rKrishna” (Text before \r is not printed)
* #echo “Hello”>file1.txt (The string will be stored in the file)

1. **whoami**

* #whoami (Provides basic info that is extremely useful when working on multiple systems)

1. **sort**

* #sort filename (To sort the output of the file)

1. **df**

* #df -h (Gets the details of the filesystem)

1. **wc : Word Count**

* #wc file\_name (Prints the following (6 10 83 file\_name) the first one refers No.of lines , No.of words and Character Count)
* #wc -w file\_name (Shows the number of words)
* #wc -l file\_name (Shows the number of lines)
* #wc -m file\_name (Shows the number of characters in a file)

1. **nl**

* #nl file\_name (Gives the line numbers for each line)

1. **head**

* #head file\_name (Prints the top 10 lines in a file)
* #head -n 10 file\_name (Prints specified(10) no.of lines only from the top)
* #head -c 10 file\_name (Prints the specified(10) no.of bytes only from the top)
* Ex: ls -t | head -n 3

1. **tail**

* #tail file\_name (Print the last 10 lines in a file)
* #tail -n 5 file\_name (Prints specified(5) no.of lines from bottom)
* #tail +5 file\_name (Prints the data starting from the specified(5) line number)
* #tail -c 5 file\_name (Prints the specified(5) no.of bytes from the bottom)
* ~~To print a specific line of data in file ( #head 13 file\_name|tail +13 ) it prints only the 13~~~~th~~ ~~line.~~
* To print a specific line of data in file, we can use ( **#sed -n ‘13p’ file\_name**) to print the 13th line only.(SED: stream editor)
* To display from the range of line ( #head -30 file\_name|tail +20) it displays from 20th to 30 lines of data. Or **(#sed -n ’20,30p’ file\_name**).
* To display range of lines in a file (#sed -n -e '25,30p' -e '32,34p' numbers)

1. **vi editor**

* #vi file\_name (You entered into the editor with the specified file\_name)
* Press “i” to write the content into the file.
* Press “Escape” + “:” + “wq” to save and exit from the editor , or press “q” for exit only without saving the changes.

**Network Commands**

1. **ping**

Sends Internet Control Message Protocol (ICMP) Echo request messages in the form of packets to the destination computer and waits in order to get the response back.

* #ping google.com or ip address

1. **nslookup**

It queries the DNS in order to fetch the IP address or the domain name from DNS records.

* #nslookup google.com/facebook.com

1. **traceroute**

To determine the path along which a packet travels. It also returns the no.of hops taken by the packet to reach the destination.

* #traceroute google.com

1. **host**

To find the domain name associated with the IP address or vice versa.

* #host google.com or IPaddress(142.250.76.110)

1. **netstat**

Used to display routing tables, connection info and the status of ports.

* #netstat

1. **arp (Address Resolution Protocol)**

Used to display and modify ARP cache. Which contains the mapping of IP address to MAC address.

* #arp

1. **Ifconfig (Interface Configuration)**

It is a utility in an operating system that is used to set or display the IP address and netmask of a network interface.

* #ifconfig

1. **Dig (Domain Info Gropper)**

Used to find the query info related to domain name and troubleshoot DNS issue in linux.

* #dig A google.com

1. **route**

Display and manipulate the routing table in linux. It contains how the network packets should be routed through a network.

* #route

**Disk Utilities**

1. **du (Disk Usage)**

It helps you to find out the disk usage of set of files or a directory.

* #du filename
* #du -c filename/dir\_name (Displays the size in grand total)
* #du -h filename/dir\_name (Prints the size in human readable format)
* #du -s filename/dir\_name (Prints the total disk space used by a specific directory or file)

1. **df (Disk Free)**

We can check the used and available disk space in linux system.

* #df
* #df -h (Shows in human readable format)
* #df -T (To display file system types)
* #df -i (To display inode usage)
* #pydf (Advanced command)
* #top (To see the all running processes in the file system) (Press ctrl + c or q to exit)

1. **fdisk**

It is most commonly used to check the partitions on a disk.

* #sudo fdisk -l

1. **sfdisk**

Similar to fdisk but with more features. It can display the size of each partition in MB.

* #sfdisk -l

1. **cfdisk**

It is a linux partition editor with an interactive user interface. It will be used to list out the existing partitions as well as create or modify them.

* #sudo cfdisk /dev/sdb

**Creating a user and assigning a password to it**

1. To create a user

* #useradd -u id\_name -g group\_id -m -d /home/user\_name -c “Comment about the user” -s /bin/bash user\_name
* To validate a user (#id user\_name)
* To assign a password (#passwd user\_name)
* To switch from one user to another user (#su – user\_name)
* To see all the password related information (#cat /etc/shadow)
* To edit the password settings (#chage user\_name)
* To lock the user (#usermod -L user\_name)
* To unlock the user (#sudo usermod -U user\_name)

1. **Setuid**

* It is a special permission file that applies to the executable file, so that the user can able to run those files with the owner permissions
* (#chmod 4644 file\_name)

1. **Setgid**

* It is also same as setuid, but instead of user it applies to the whole group.
* #chmod 2644 file\_name

1. **Stickybit**

* Even though all the users having the full rights on the directory but only the admin can only remove the directory.
* #chmod +t dir\_name (To remove it use -t)
* The default stickybit directory is (/tmp)

**Creating a disk partition**

* Create a EBC volume of same region, how much GB you want.
* #fdisk /dev/xvdf(disk\_name)
* Press m for details
* Press n (To add a new partition)
* Select(default p) :p
* Partition number(1-4,default 1) : 1 (select how much partitions you need)
* Size : +50G
* Press w (To save the partition and exit)
* #partprobe /dev/xvdf (The server will restart and it is mounted to the filesystem)
* #mkfs.ext4 /dev/xvdf (To create a filesystem)
* #mount /dev/xvdf /directory\_name(a new directory) (To mount the directory to the disk)
* #df -h (To verify it)
* To make the mount system permanent on the server
* #vi /etc/fstab
* Enter the following under UID

#/dev/xvdf /directory\_name defaults 1 1

#mount -a (To verify)

**Users in Linux**

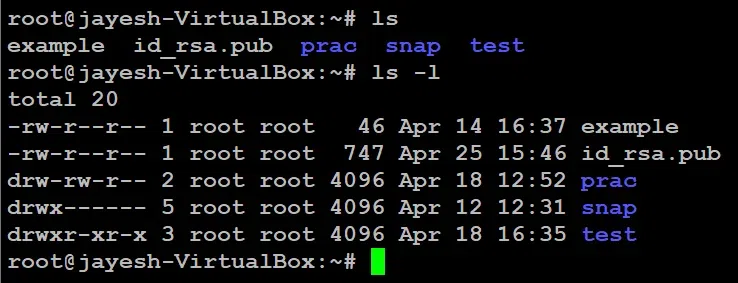
* #awk -F':' '{ print $1}' /etc/passwd (To list out all the users in Linux, use the awk command with -F option. Here, we are accessing a file and printing only first column with the help of print $1 and awk.)
* #id <username> (Using id command, you can get the ID of any username. Every user has an id assigned to it and the user is identified with the help of this id.)
* #sudo useradd <username> (useradd command adds a new user to the directory.)
* #passwd username (Using passwd command to assign a password to a user. After using this command we have to enter the new password for the user and then the password gets updated to the new password.)
* #userdel -r username (You can also delete a user name. The below command deletes the user whose username is provided.)

**Groups in Linux**

In Linux, **groups are collections of users**. Creating and managing groups is one of the simplest ways to deal with multiple users simultaneously, especially when dealing with permissions. The /etc/group file stores group information and is the default configuration file.

* #groups [username] (To view the group memberships of a user is the most common usage of the `groups` command.)
* #groups (To display group membership for the current user)

**File Permissions**



There’s a lot of information in those lines.

* The first character = ‘-‘, which means it’s a file
  + - ‘d’, which means it’s a directory.
* The next nine characters = (rw-r–r–) show the security
* The next column shows the owner of the file. (Here it is `root`)
* The next column shows the group owner of the file. (Here it is `root` which has special access to these files)
* The next column shows the size of the file in bytes.
* The next column shows the date and time the file was last modified.
* Last Column = File\_name or Directory\_name. (For example, here are: prac, snap, test, example)

**Security permissions in Linux**

First, you must think of those nine characters as three sets of three characters (see the box at the bottom). Each of the three “rwx” characters refers to a different operation you can perform on the file.

--- --- ---

rwx rwx rwx

user group other

Changing security permissions

The command you use to change the security permissions on files is called “chmod“, which stands for “change mode” because the nine security characters are collectively called the security “mode” of the file.

* #chmod o+x xyz.txt

user -> u

group -> g

others -> o

all -> a

* #chmod ugo-rwx xyz.txt (if you want to take all permissions away from everyone)
* #chmod 777 <file\_name> (provide full read write and execute permission)

**Archive and Compression in Linux**

Archiving involves combining multiple files or directories into a single file, making it easier to transfer or store them. Compression, on the other hand, reduces the size of files to save disk space and speed up data transfer. Two commonly used tools for archiving and compression in Linux are tar and zip .

**tar**

tar stands for Tape Archive and is a command-line utility in Linux used for archiving files and directories. It does not perform compression by itself but is commonly used in combination with compression tools like gzip, bzip2, or xz to create compressed archives. tar is widely used for creating backup files and distributing software packages.

**zip**

zip is another command-line utility used for both archiving and compressing files and directories in Linux. Unlike tar, zip supports built-in compression, so you can create compressed archives directly without the need for additional compression tools. It's often used for sharing files between different operating systems and in Windows environments.

**To create a new archive with tar, use the following command:**

#tar -cvf archive\_name.tar file1 file2 directory1

* -c: Create a new archive.
* -v: Verbose mode, showing the progress of archiving.
* -f: Specifies the output filename (archive\_name.tar in this case).

**To extract files from a tar archive, use this command:**

#tar -xvf archive\_name.tar

Adding Files to an Existing Archive

**To add files to an existing tar archive, use this command:**

#tar -rvf archive\_name.tar file3 file4

**Compression with tar**

tar can work in combination with various compression tools to create compressed archives.

**Compressing with gzip**

To create a tar archive with gzip compression, use the following command:

#tar -czvf archive\_name.tar.gz file1 file2 directory1

* -c: Create a new archive.
* -z: Compress the archive with gzip.
* -v: Verbose mode, showing the progress of archiving.
* -f: Specifies the output filename (archive\_name.tar.gz in this case).

(**bzip2, xz)**

**Archiving and Compression with zip**

**Creating a ZIP Archive**

To create a new ZIP archive with zip, use the following command:

#zip archive\_name.zip file1 file2 directory1

**Extracting from a ZIP Archive**

To extract files from a ZIP archive, use this command:

#unzip archive\_name.zip

Adding Files to an Existing ZIP Archive

To add files to an existing ZIP archive, use this command:

#zip -r archive\_name.zip file3 file4

**Password Protection in ZIP Archives**

You can password-protect a ZIP archive by using the -P flag followed by the password:

#zip -r -P password123 secure\_archive.zip private\_file1 private\_directory

Remember to replace password123 with your desired password.

**Package Installation**

To install a package using the apt command, you will need to use the following syntax: apt install package\_name. The “apt” command can be used to install software packages from a software repository.

**Linux Filesystem Hierarchy Standard (FHS)**

It explains where files and directories should be located and what it should contain.(#ls /)

**SED Command**

SED command in UNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace.

Replacing or substituting string : Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

$sed 's/unix/linux/' geekfile.txt

Here the “s” specifies the substitution operation. The “/” are delimiters. The “unix” is the search pattern and the “linux” is the replacement string.

**For root user (#apt update)**

**For ubuntu user(#sudo apt update)**

**To install java in ubuntu on root user (#apt install openjdk-11-jdk)**

**To connect the bastion host to private instances , for copying the pem file from local machine to linux server(#scp -i C:\Users\lenovo\Downloads\aws-login.pem Cscp -i C:\Users\lenovo\Downloads\aws-login.pem C:\Users\lenovo\Downloads\aws-login.pem [ubuntu@13.239.117.19:/home/ubuntu](mailto:ubuntu@13.239.117.19:/home/ubuntu))**

**To connect to the private instance in bastion host use(#ssh -i aws-login.pem [ubuntu@10.0.137.131](mailto:ubuntu@10.0.137.131))**

To Update the server: #sudo apt update //for Ubuntu distribution

**VI Editor**

In command mode(VI) only HJKL keys only work.

Command Mode (esc :)

Insert mode (i)

Executable mode (esc )

To save the file (Press Esc :wq!) ‘w’ is for save , ‘q’ is for quit and ‘!’ is for force quit

H from current position to left

L from current position to right

J from current position to down side

K from current position to up side

To go to the end of the line press I

To go to the beginning of the line press A

If we want to see the context with line numbers in a VI editor go to executable mode i.e, (Esc:) and enter “set nu” I.e, (:set nu), to hide the line numbers (:se no nu)

Place the cursor on a 4th line and you want to copy the content from 4th line to 9th line then press ‘5YY’

Use ‘shift G ‘ to go to the end of the line .

And press ‘p’ to paste the content.

If we want to delete a line press ‘dd’

To delete multiple line hold the number for example 4 and press ‘dd’ then it will delete last 4 lines in the file.

If you want to undo the changes you made just press ‘u’

If we want to remove the directory containing the files in it, but we have to remove those directory from the file system we use

#rm -rf <file\_name>

R means recursively

F means forcefully

**Shell Scripting**

**Definition:** It accepts commands as input from the users and execute them. Sometimes we want to execute a bunch of commands routinely, so we have to type in all commands each time in the terminal.

We can write all these commands in a file and can execute them in shell to avoid repetitive work. These files are called shell scripts. Each shell script is saved with the ‘.sh’ extension.

**Why do we need shell scripts?**

-> To avoid repetitive work and automation.

-> System admins use shell scripts for routine backups.

-> System Monitoring.

**Advantages:**

-> The commands and syntax are exactly the same as those directly entered in the command line.

-> Writing shell scripts are much quicker.

-> Quick start.

**Disadvantages:**

-> Slow execution speed.

-> Not well suited for large and complex task.

**First shell script :**

$ vi first.sh

#!/bin/bash /\*Shebang line (which tells the OS to use which interpreter for executing the commands.)\*/

date

touch demo1

ls -l

Save the above file .

After saving give the execute permission for the file.

Chmod +x first.sh

To execute the shell script

sh first.sh /\* It creates a new shell process and executes. \*/

Or

./first.sh /\* It executes in the current shell \*/

**Ex 2: Create a new file if it exists it won’t create it**

$vi file.sh

#!/bin/bash

If [ -f demo1 ] /\* we use -f for file checking and -d for directories \*/

then

echo “demo1 file already exists”

else

touch demo1

echo “demo1 file created”

fi

**Passing the parameters instead of 1 file name as**

Ex: $vi file2.sh

#!/bin/bash

if [ -f $1 ]

then

echo "$1 file is already exists."

else

touch $1

echo "$1 file is created."

fi

/\* $1 is for the argument which is passed after the filename at the time of execution. \*/

$sh file2.sh demo1

If we want to know whether the previous command is successfully executed or not.

$echo $?

If it returns 0 then the command successfully executed.

**Ex 3: Read the input file name from the user and if the file is there return some text or create it.**

$vi file3.sh

#!/bin/bash

echo "Please enter a file name":

read file

if [ -f $file ]

then

echo "$file file is already exists."

else

touch $file

echo "$file file is created now."

fi

$sh file3.sh

Please enter a file name:

Demo2

Demo2 file is created now.

**To find out the which shell you’re using now**

$echo $SHELL

$echo $USER

**Create a user in the root account if it is not root then don’t create it, if it is create a user if already exists the user simply exit**

Ex:

#!/bin/bash

if [ $(id -u) -eq 0 ];then

read -p "Enter Username: " username

egrep "^$username" /etc/passwd > /dev/null

if [ $? -eq 0 ];then

echo "User already exists."

exit 1

else

useradd -m $username

echo "$username is created now"

fi

else

echo "Only root may add a new user to this server."

fi

To check whether the user is created or not

$id <username>

**Passing the arguments and printing**

Ex: vi args.sh

#!/bin/bash

echo "$1"

echo "$2"

echo "$3"

echo "$4"

Execution: root@ip-172-31-213-198:~/shellscr# sh args.sh a b c d

a

b

c

d

**Pass atleast 1 argument or else fail the exection**

Ex: vi args1.sh

#!/bin/bash

if [ "$#" -ge 1 ]

then

echo "$1"

echo "$2"

echo "$3"

else

echo "Please enter at least 1 argument while executing the script."

fi

**Execution:** sh args1.sh a b c d

a

b

c

d

$sh args1.sh

Please enter at least 1 argument while executing the script.

**To add multiple users at a time.**

First create a file consists of all usernames as

$vi users

demo11

demo22

demo33

demo44

demo55

Then create shell script as

$vi forusers.sh

#!/bin/bash

for user in `cat users`

do

useradd $user

done

**In Shell Script Comments are written as ‘#’**

$ #this is a comment

**How do you troubleshoot the shell script file if it is failed the execution.**

$ sh -x for.sh