Linux File Hierarchy Structure / Linux Directory Structure

Linux is a single root directory system.

/ is "the parent folder". Also called the root directory.

1. / (Root):
    - /root is the root user's home directory, *which is not the same as /*
    - Only the root user has the right authority to write under this directory.
    - Every single file and directory starts from the root directory.
    - It is the Primary hierarchy root and root directory of the entire file system hierarchy.

2. /bin :
    - It contains all the binaries executables (all the commands which are there in your Linux machine) that need to be available for all users, e.g., cat, ls, cp, ps, ping, grep, rm, mkdir, touch
    - Common Linux commands you need to use in single-user modes are located under this directory.

3. /dev :
    - These include hardware device drivers, USB, or any device attached to the system.
    - *Example: /dev/tty1, /dev/usbmon0*

4. /home :
    - Whenever a user is created in Linux then the folder with the name same as that of the username is created inside the home directory.
    - It contains the user's home directories, containing saved files, personal files/settings, etc.
    - *Example: /home/neha*

5. /lib64 :
    - It contains all the 64-bit libraries.

6. /mnt :
    - It contains the mount files.
    - *Mount files are the files that are accessible and attached to an existing directory structure.*

7. /proc :
    - It contains information about the system processes (running process with a particular process ID).

8. /run:
    - It contains information about the kernel.
    - *It stores volatile runtime data. Volatile memory is the type of memory in which data is lost as it is powered off.*

9. /srv :

- It contains information about the services.
- It contains site-specific data served by the system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.

10. /tmp :

- It contains information about the temporary files created by the system and users.
- *Files under this directory are deleted when the system is rebooted.*

11. /var :

- It contains variable data. Variable data is that data that gets changed every time eg. log files because the content of the log files is constantly changing.

12. /boot :

- Here the boot sequence of the machine is present. Here the information of how the machine will boot up is present.
  All the files required for booting up the machine are present in the boot directory.
- If by any chance this folder gets deleted then your machine won't be able to boot up.
- It contains Linux bootable files,
- It contains boot loader (GRUB)

13. /etc :

- It contains all the system configuration files in Linux.
- Contains configuration files required by all programs.

14. /lib :

- It contains the library files which are not 64 bits.

15. /media :

- It contains all the removable devices that you insert into your machine eg. CD, Pendrive

16. /opt :

- All the third-party software is present under this directory.

17. /sbin :

- It contains all the supervisory executable binaries (all the commands which are there in your Linux machine) used by the superuser.
- These are special binaries. Eg. usermod, useradd, usergroup

18. /sys :

- It contains all the system files.

19. /usr :

- If a user creates any software or the user custom software is present in the user directory.

Commands in Linux

su -> switch user
su - root
su - <username>

pwd -> print working directory (to get the absolute path)

~ -> represents the home location of the current user

useradd <username> -> to add or create users accounts to your system
passwd <username> -> to add or change passwords to users accounts

userdel <username> -> to delete users accounts from your system
userdel -r <username> -> -r stands for recursive (meaning, also the home directories are removed)

mkdir <foldername> -> create new folder
mkdir -p <new folder path to be created> -> create new folder with the specific path
Eg. mkdir -p songs/hindi/gazals

ls -> used to view a list of files and folders in a given directory

cd <new location> -> to change directory to a new location

touch <file name> -> to create a new file without any content
touch file{1..10} -> to create many new files without any content using looping

cat <file name> -> used to read data from the file and gives their content as output

rm -rf <file/folder> -> rf stands for recursive forcefully
rm -rf * -> remove all the files

cd ../../

cd ../../ ; ls -> combining two commands using colon ;

To ways to edit files in the file editor: vi and nano

vi <file name>

Press i to insert data into the file. Now you are in insert mode.

Add the content

To save and exit -> press esc + :wq! (here,! means forcefully)

If you don't want to write and only want to view the content and exit -> press esc + :q!

nano <file name>

It is already in insert mode. Add the content.

Save & exit -> ctrl + x -> y

Current & parent location with cd

cd . -> staying in the name location

cd .. -> moving to parent location

Copy paste

cp <source file name> <destination file name> -> if both files are in the same location

cp <source file name path> <destination file name path> -> if both files are in different location

cp * -> copy all files in the directory

cp file? <destination path> -> ? maps only one character

cp file1* <destination path> -> * maps all the characters

cp file*0 <destination path> * maps all the characters

Cut paste

mv <source file name> <destination file name> -> if both files are in the same location

mv <source file name path> <destination file name path> -> if both files are in different location

mv <source file name path>* <destination file name path> -> if more than 1 files are present in the source path

---

Renaming files/folders

mv <old name> <new name>

To fetch IP from the DORA process if DHCP is available.

dhclient -v -> v means verbose means tell me what is happening

---

To check if we have access to internet

nslookup <google.com>

nslookup <8.8.8.8> -> it is the public IP for google

To add a new group

groupadd <group name> -> to create a new group

groupmod -n <new groupname> <existing groupname>

To add a new user to the group
useradd -g <group name> <username>

To add an existing user to the group
usermod -g <group name> <username>

The user information is stored in file -> /etc/passwd
The file that stores user's actual password in encrypted format is -> /etc/shadow

---

The group information is stored in file -> /etc/group
The file that stores shadowed information about the group accounts is -> /etc/gshadow

---

[Users in Linux]
[username@machinename <curent working location>]$

$ --> indicates that you are working as a non-root user
# --> indicates that you are working as a root user

There are 3 types of users in Linux:

1. Root (All possible privileges)
2. Superuser (sudoer)
3. Regular user

Some key points

- All the information about users in Linux is stored in '/etc/passwd' file
- Whenever you create a new user, a folder with the same name as that of username, should be created under /home
- Password should always be assigned to a newly created user
- Password information is stored in '/etc/shadow'
- Syntax to switch the users is 'su - <username>'
- Whenever you create a new user, a group with the same name as that of username, is also created
- When you delete a user from a Linux machine, its folder under /home should also be deleted
- 'root' can make any user a superuser
- The information about super users or sudoers stay in '/etc/sudoers' file
- Regular users in Linux get the user id >= 1000 - Explanation?

User creation & its verification:

useradd <username>
passwd <username>

User should present in /etc/passwd -> cat etc/passwd
User entry should be present in /etc/home -> ls /etc/home

Password information

cat /etc/shadow
1st field is username, 2nd field is the encrypted password

Group information -> cat /etc/group

---

su significance -> defines the environment of users

su - neha ---> /home/neha
su neha ---> /root

To make a user sudoer

Vi /etc/sudoers
Add the user with ALL=(ALL) permissions
sudo useradd <username>
sudo passwd <username>
sudo userdel -r <username>

User ID’s

Root UID is 0
All other users is >= 1000
System or service users UID is 1 - 999

The parameter for minimum user id that can be allocated to any user is defined in -> /etc/login.defs

---

File permissions in Linux

		Numeric representation	
read	r	4	open and read a file
write	w	2	modify the contents of a file (add, remove and rename files)
execute	x	1	Execute a file

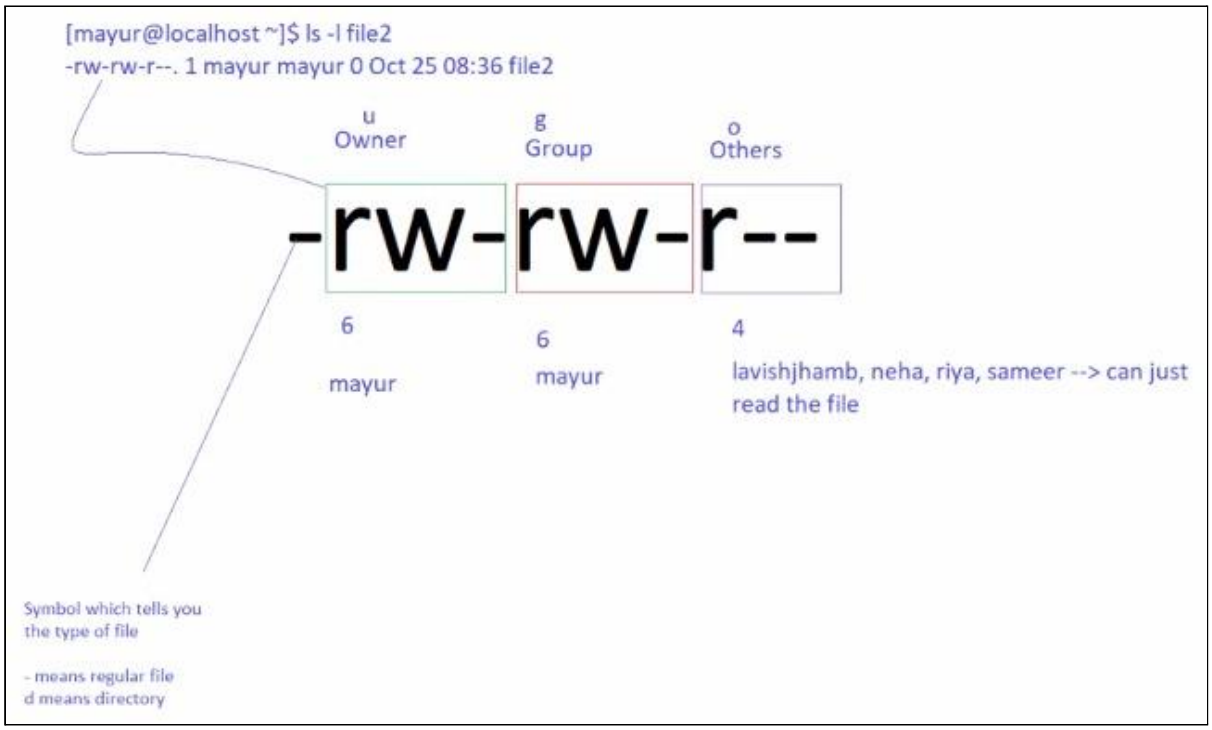
Full control -> rwx or 7

There are 3 attributes of users for each file:

1. One who creates a file i.e owner of the file.
2. Second is the group to which the owner belongs.
3. Rest of the users in Linux.

To see permissions of a file
ls -l <filename>
- symbol which tells you the type of file.
- means a regular file
d means directory

cat /etc/passwd | tail -n 5



To modify the permissions of the file
chmod 755 <filename>

Providing permissions using + and removing permissions using -
chmod o+w <filename> ---> here o means others, & w means write
chmod o-w <filename> ---> here o means others, & w means write

chmod 646 <filename> ---> means rw-r---rw-
chmod 75 <filename> is same as chmod 075 <filename>

To change the ownership of the file

chown <owner>:<group owner> <filename>
Eg. chown neha:neha file2

chown <owner>:<group owner> <directory>
Eg. chown neha:neha /home/mayur/file2

ls -l <filename>

The owner cannot change ownership of the file, ONLY root can !

chown <new owner username> <filenameor directory>
chgrp <new owner groupname> <filenamor or directory>

Access control list
ACL, these are used in special cases when access needs to be granted to specific users & not all.

To set ACL
setfacl -m u:<username>:<permission set> <filename>
setfacl -m u:<username>:<permission set> <directory>

ls -l <filename>

To view the permissions set to the users ACL
getfacl <filename>

---

useradd -g <exisiting usergroup type> <new user's username>
Eg. useradd -g nikhil tom
su - tom
ls -l

UMASK

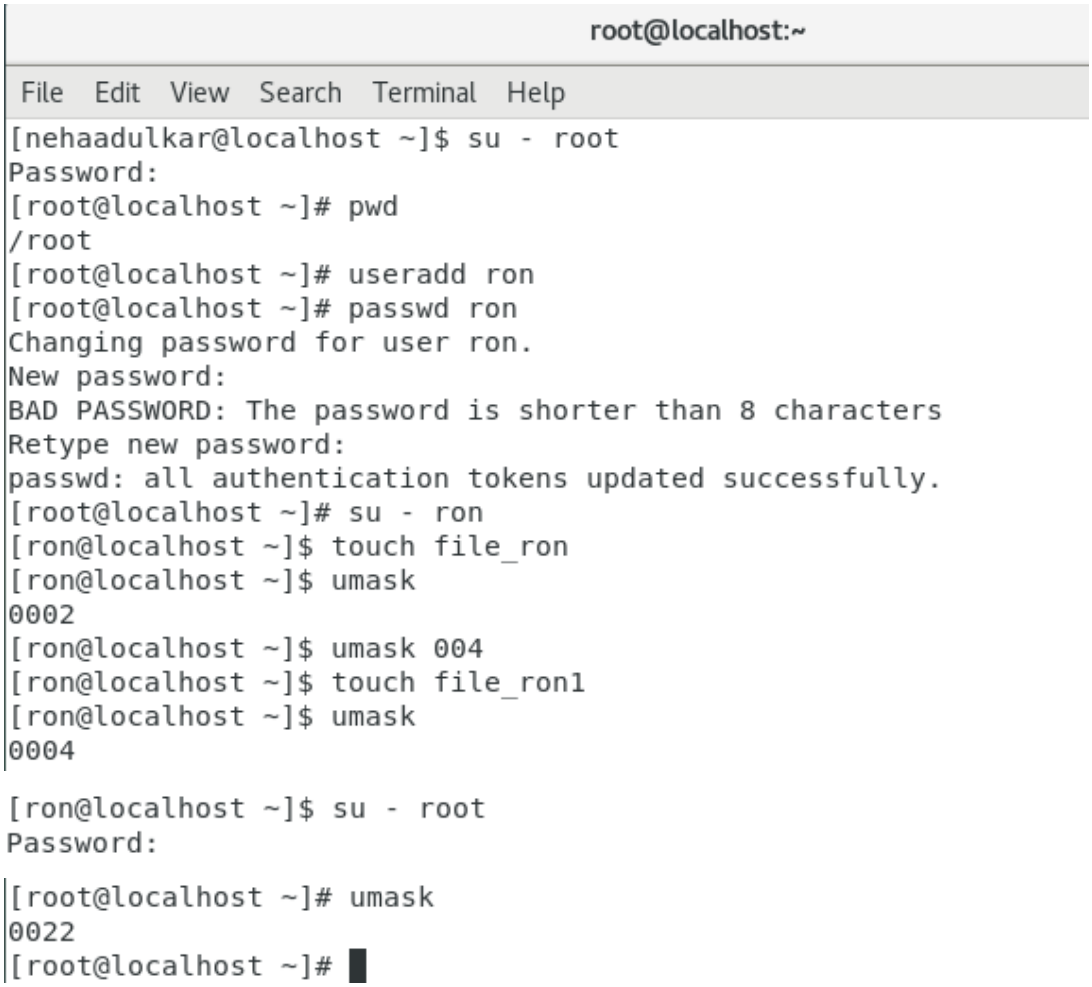
It is a variable defined by shell. The value of this variable decides the default permissions on the newly created files or directories.
To see the umask value command is ---> umask
Root has umask 022 & other users have umask as 002

By default, the value of the directory is 777 & the file is 666.
To define the effective permissions, umask is subtracted & then effective permissions of the file/directory are calculated.

Eg. I got the permissions of the file as 644. So, what happened was 666 - 002 = 644

To see the umask after subtraction i.e to view the permissions of file in numeric form command is --->
stat -c %a <filename>

Root's umask is more restrictive.



Sticky Bits (T)

A Sticky bit is a permission bit that is set on a file/directory that allows only the owner of the file/directory or the root user to delete or rename the file. No other user is given privileges to delete the file created by some other user.

Without the sticky bit set, any user with write and execute permissions for the directory can rename or delete contained files, regardless of the file's owner.
Eg. Google drive

//create 3 users under editiss folder
[nehaadulkar@localhost ~]$ su - root
Password:
[root@localhost ~]# mkdir editiss
[root@localhost ~]# cd editiss
[root@localhost editiss]# useradd ron
[root@localhost editiss]# passwd ron
[root@localhost editiss]# useradd potter
[root@localhost editiss]# passwd potter
[root@localhost editiss]# useradd granger
[root@localhost editiss]# passwd granger

//view the users
[root@localhost editiss]# cat /etc/passwd | tail -n 4
nehaadulkar:x:1000:1000:NehaAdulkar:/home/nehaadulkar:/bin/bash
ron:x:1001:1001::/home/ron:/bin/bash
potter:x:1002:1002::/home/potter:/bin/bash
granger:x:1003:1003::/home/granger:/bin/bash

//create a folder drive
[root@localhost editiss]# mkdir drive

//set ACL permissions to the 3 users with full (rwx) permissions
[root@localhost editiss]# setfacl -R -m u:ron:rwx /root
[root@localhost editiss]# setfacl -R -m u:potter:rwx /root
[root@localhost editiss]# setfacl -R -m u:granger:rwx /root

//login using ron user and create a file_ron
[root@localhost editiss]# su - ron
[ron@localhost ~]$ cd /root/editiss/drive
[ron@localhost drive]$ touch file_ron
[ron@localhost drive]$ logout

//login using potter user and create a file_potter
[root@localhost editiss]# su - potter
[potter@localhost ~]$ cd /root/editiss/drive
[potter@localhost drive]$ touch file_potter
[potter@localhost drive]$ logout

//login using granger user and create a file_granger
[root@localhost editiss]# su - granger
[granger@localhost ~]$ cd /root/editiss/drive
[granger@localhost drive]$ touch file_granger
[granger@localhost drive]$ logout

//list down the files
[root@localhost editiss]# ls drive
file_granger file_potter file_ron

//login using ron and delete potter user's file
[root@localhost editiss]# su - ron
Last login: Tue Oct 26 20:03:02 IST 2021 on pts/0
[ron@localhost ~]$ cd /root/editiss/drive
[ron@localhost drive]$ rm -rf file_potter
[ron@localhost drive]$ logout

//view the files in the directory (potter's file is missing
[root@localhost editiss]# ls
drive
[root@localhost editiss]# cd drive
[root@localhost drive]# ls
file_granger file_ron

//apply sticky bits to the drive folder
[root@localhost drive]# chmod +t /root/editiss/drive

//login using ron user and view the list of files
[root@localhost drive]# su - ron
Last login: Tue Oct 26 20:04:02 IST 2021 on pts/0
[ron@localhost ~]$ ls /root/editiss/drive
file_granger file_ron

[ron@localhost ~]$ pwd
/home/ron
[ron@localhost ~]$ cd /root/editiss/drive

//using ron user try to delete granger's file and the operation is not allowed due to the concept of the sticky bits

[ron@localhost drive]$ rm -rf file_granger

rm: cannot remove 'file_granger': Operation not permitted

//view the list of files

[ron@localhost drive]$ ls

file_granger file_ron

[ron@localhost drive]$ ls -l

total 0

-rw-rw-r--. 1 granger granger 0 Oct 26 20:02 file_granger

-rw-rw-r--. 1 ron   ron   0 Oct 26 20:01 file_ron

//to check if sticky bits are applied or not. (T symbol)

[ron@localhost drive]$ ls -ld

drwxrwxr-t+ 2 root root 42 Oct 26 20:04 .

[ron@localhost drive]$

---

grep

grep is used to print lines on the basis of a specific word or a pattern.

Disk Partitioning in Linux

Reference: https://www.geeksforgeeks.org/disk-partitioning-in-linux/

---

cat /etc/passwd | grep bash

```
[root@localhost ditiss]# cat /etc/passwd | grep bash
root:x:0:0:root:/root:/bin/bash
nehaadulkar:x:1000:1000:NehaAdulkar:/home/nehaadulkar:/bin/bash
```

Description of fields in etc/passwd

It has 7 fields:

1. Username
2. x ---> here x has the link to the password file, etc/shadow
3. UserID
4. GroupID
5. Comment
6. Home location
7. The shell that the user is using.

Types of shell → sh & bash

---

cut

cat /etc/passwd | grep <shell type> | cut -d'<field separator>' -f<field number>

Q. Give a list of users.

cat /etc/passwd | cut -d':' -f1

```
[root@localhost ditiss]# cat /etc/passwd | cut -d':' -f1
root
bin
daemon
adm
```

Q. Give a list of users (who are using the bash shell).

cat /etc/passwd | grep bash | cut -d':' -f1

```
[root@localhost ditiss]# cat /etc/passwd | grep bash | cut -d':' -f1
root
nehaadulkar
```

Q. Give a list of users (who are using the bash shell) along with their userids.

cat /etc/passwd | grep bash | cut -d':' -f1,3

```
[root@localhost ditiss]# cat /etc/passwd | grep bash | cut -d':' -f1,3
root:0
nehaadulkar:1000
```

---

awk

Q. Give a list of users (who are using the bash shell) along with their userids.

cat /etc/passwd | grep bash | awk -F':' '{print $1,$3}'

```
[root@localhost ditiss]# cat /etc/passwd | grep bash | awk -F':' '{print $1,$3}'
root 0
nehaadulkar 1000
```

Q. Fetch the list of all users where userid is greater than 1000 and display any username & userids.

cat /etc/passwd | grep bash | awk -F':' '{if ($3 > 1000) print $1,$3}'

```
[root@localhost ditiss]# cat /etc/passwd | grep bash | awk -F':' '{if ($3 > 1000) print $1,$3}'
tom 1001
[root@localhost ditiss]#
```

Usage of chage command:

The chage command changes the number of days between password changes and the date of the last password change. This information is used by the system to determine when a user must change their password.

chage -l <useranme>

chage -l tom

```
[root@localhost ditiss]# chage -l tom
Last password change                                    : Oct 28, 2021
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
[root@localhost ditiss]#
```

chage -m 1 -M 90 tom
chage -l tom

```
[root@localhost ditiss]# chage -m 1 -M 90 tom
[root@localhost ditiss]# chage -l tom
Last password change                                    : Oct 28, 2021
Password expires                                        : Jan 26, 2022
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 1
Maximum number of days between password change          : 90
Number of days of warning before password expires       : 7
```

The chage command expires the user's password 90 days from the last password change.

Shell-scripting

Q.1 Create 5 directories and 5 files

[root@localhost ditiss]# mkdir scripts
[root@localhost ditiss]# cd scripts
[root@localhost scripts]# ls
[root@localhost scripts]# vi task1.sh → open the editor
[root@localhost scripts]# chmod +x task1.sh → gives executable permissions to file
[root@localhost scripts]# ls -l task1.sh
-rwxr-xr-x. 1 root root 46 Oct 28 18:32 task1.sh
[root@localhost scripts]# ./task1.sh → run the script

```
[root@localhost scripts]# ls
dir1  dir2  dir3  dir4  dir5  file1  file2  file3  file4  file5  task1.sh

[root@localhost scripts]# cat task1.sh
#!/bin/bash
mkdir dir{1..5}
touch file{1..5}
```

Q.2 Create 5 directories and 5 files.
dir1 should contain file1, dir2 should contain file2, and so on ..

```
[root@localhost scripts]# vi task2.sh
[root@localhost scripts]# chmod +x task2.sh
[root@localhost scripts]# ./task2.sh
[root@localhost scripts]# ls
dir1  dir2  dir3  dir4  dir5  task2.sh
[root@localhost scripts]# cd dir1
[root@localhost dir1]# ls
file1
[root@localhost dir1]# cd ..

[root@localhost scripts]# cat task2.sh
#!/bin/bash
for i in {1..5}
do
mkdir dir$i
cd dir$i
touch file$i
cd ..
done
```

Q.3 Find list of users on Linux m/c whose userid is >= 1000 and display in following format
username uid shell

```
#!/bin/bash
cat /etc/passwd | awk -F':' '{if ($3 >= 1000) print $1, $3, $7}'

[root@localhost scripts]# ./task3.sh
nfsnobody 65534 /sbin/nologin
nehaadulkar 1000 /bin/bash
tom 1001 /bin/bash
ron 1002 /bin/bash
```

```
#!/bin/bash
cat /etc/passwd | awk -F':' '{if ($3 >= 1000) print $1, $3, $7}' | grep -v nfs

[root@localhost scripts]# ./task3.sh
nehaadulkar 1000 /bin/bash
tom 1001 /bin/bash
ron 1002 /bin/bash
```

Taking user inputs for scripting → **read** command is used to take user input

read -p “content” <variable name>

Q.4 Create a script that asks for a username & displays its userid.

```
#!/bin/bash
read -p "Enter a username: " var
echo "Userid for $var is as follows: "
cat /etc/passwd | grep "$var" | cut -d':' -f3


[root@localhost scripts]# ./task4.sh
Enter a username: tom
Userid for tom is as follows:
1001
```

Storing the output of the command in some variable → user defined variable

<variable name> = $(<command>) → Recommended method
<variable name> = `<command>`

```
#!/bin/bash
tomuid=$(cat /etc/passswd | grep tom | cut -d':' -f3)
echo $tomuid
```

Q.5 Create a script that asks for a username & displays its userid only if it is > 1002

```
#!/bin/bash
read -p "Enter username: " name
uid=$(cat /etc/passwd | grep "$name" | awk -F':' '{print $3}')
if [[ $uid > 1002 ]]
then
echo "UID for $name: $uid"
else
```

```
echo "UID is <= 1002, hence not printing it"
fi

[root@localhost scripts]# ./task5.sh
Enter username: tom
UID is <= 1002, hence not printing it
[root@localhost scripts]# ./task5.sh
Enter username: granger
UID for granger: 1003
```

Q6. Create a script that asks for a word. Create a directory with the name as that of input. Inside that directory create the following,
dir1 → file1 → I'm file1
dir2 → file2 → I'm file2, and so on ..

```
#!/bin/bash
read -p "Enter a word: " var
mkdir $var
cd $var
for i in {1..1000}
do
mkdir dir$i
cd dir$i
touch file$i
echo "I'm file$i" >> file$i
cd ..
done
```

Q7. Write a shell script to get the current date-time, username & current working directory.

```
#!/bin/bash
echo "Date & time: "
date
echo "Username: "
whoami
echo "Current working directory: "
pwd

[root@localhost scripts]# ./task7.sh
 Date & time:
Thu Oct 28 22:38:51 IST 2021
Username:
root
```

Current working directory:
/root/ditiss/scripts

Q8. Create a script that asks for a user name displays the user ID of the user-provided as input.
If the user does not exist then show the following error, "Entered user is not present in your Linux machine". And if the user is present also mention the which shell, user is using.

```
#!/bin/bash
read -p "enter username: " usr
var=$(cat /etc/passwd |grep "$usr" | awk -F':' '{print $3}')
if [[ $var =~ [0-9] ]]
then
   echo "$usr exists"
   echo "UID for $usr: $var"
   shell=$(cat /etc/passwd | grep "$usr" | cut -d':' -f7)
   echo "$usr is using shell: $shell"
else
   echo "$usr is not present on your Linux Machine"
fi
```

Q9. Create a script that asks for a user name and tells you its userid and tells if the ID is greater than 1000 or not.

```
#!/bin/bash
read -p "enter username: " usr
var=$(cat /etc/passwd | grep "$usr" | awk -F':' '{print $3}')
echo "UserID of $usr is: $var"
if [[ $var > 1000 ]]
then
echo "$usr ID is greater than 1000."
else
echo "$usr ID is not greater than 1000"
fi
```

Q10. Create a script that asks for a path and tells you if the entered path is a file or a dir.

```
#!/bin/bash
read -p "enter your path: " a
if [[ -d $a ]]
then
        echo "$a is a directory"
elif [[ -f $a ]]
then
```

```
        echo "$a is a file"
else
        echo "Invalid path !!"
fi
```

[root@dns scripts]# ./task10.sh
enter your path: /etc/passwd
/etc/passwd is a file
[root@dns scripts]# ./task10.sh
enter your path: /etc
/etc is a directory
[root@dns scripts]# ./task10.sh
enter your path: etc
Invalid path !!

Q11. Write a shell script that asks for a dir name and adds an extension "new" to all the files in a directory.

```
#!/bin/bash
read -p "enter the folder name: " dir
cd $dir 2> /dev/null
ls
for file in *
do
mv $file $file.new
done
```

Q12. Create a script that asks for a number between 1-7 and displays which day of the week that number represents. Assuming -1.monday