```
#!/bin/bash
# author : Surya Dev Singh Jamwal
# course : PG-DBDA
# user ID : sdevsinx
# date : 2023 Mar 21
# title : Linux Module Day03
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++

#1. id command
# id command will print User ID , Group ID, groups for the current user
id      # returns userID, groupID, groups for current user
id root     # returns userID, groupID, groups for the root, '0' is
reserved for root

#2. UID
# UID - UserID, stands for user identifier, number assigned to each user
on the system, identifies the user and determines which system resources a
user can access
```

UID Range	Purpose
UID(0)	reserved for root, since root is the first user
UID(1-99)	reserved for predefined accounts
UID(100-999)	reserved for system admins & system accounts/groups
UID(1000-9999)	reserved for application accounts
UID(>10000)	reserved for user accounts

```
#3. GID
# GID - Group ID, stands for group identifier, number assigned to each
group on the system, identifies the group and determines which system
resources a group can access
```

GID Range	Purpose
GID(0)	reserved for root group
GID(1-99)	reserved for system & application use
GID(>100)	reserved for user groups

```
#4. permissions
# There are three types of permissions :
# a. Owner permissions
# Owner permissions are used by the assigned owner of files/dirs.
# User belongs to this class.

# b. Group permissions
# Group permissions are used by members of the group who owns the
file/dir.
# A group is a collection of users.
# The main purpose of a group is to set privileges like to set read,
write, execute permissions to other users

# c. other group permissions
# Other permissions are used by all the users other than file owner,
member of the group that owns the file/dir.
#Any user who is not is a part of all the users/groups, will fall under
Others group


#5. common permissions
```

digits	permission	meaning
777	`rwxrwxrwx`	read, write, execute permission for all users
755	`rwxr-xr-x`	read & execute permission for all users, only owner has permission to write
750	`rwxr-x---`	read, write execute permission for only owner, read & execute permission for groups only, other users don't have any access
700	`rwx------`	only owner has read, write execute permission, groups & others don't have access to file
666	`rw-rw-rw-`	all have read-write permission, no-one has permission to execute the file
664	`rw-rw--w-`	read permission is only for owner & group, write permission exists for all, no-one has execute permission
644	`rw-r--r--`	only owner has read-write permission, read permission for group & others
640	`rw-r-----`	read, write permission for owner, only read permission for group

600	rw-------	only user has read-write permission, no access for group & others
400	r--------	only owner has read permission, group & others don't have write-execute permission

```
#6. rwx permission weights
```

digit	sum	read(4)	write(2)	execute(1)
7	4+2+1	r	w	x
6	4+2+0	r	w	-
5	4+0+1	r	-	x
4	4+0+0	r	-	-
3	0+2+1	-	w	x
1	0+0+1	-	-	x
0	0+0+0	-	-	-

```
#7. adding a new user
\# adduser check    # to add new user
compgen -u      # to check if user exists

#8. adding a new group
\#addgroup check1   # to add group 'check1'
getent group    # to check added groups, to check if group 'check1' is
added

#9. adding a user in a group
usermod -a -G "check1" "cdac"   # to add user 'cdac' into group 'check1'
id cdac             # to check if user 'cdac' is in group 'check1'

#10. to check owner & group of new created file
\#echo "hello" > test.txt   # create a file with permission
ls -l               # to check owner & group
su cdac             # to change user 'cdac'
ls -l test.txt          # to check owner & group of file
#chown cdac:check1 test.txt
```

```
#ls-l test.txt

#11. change permissions of a file
chmod 710 test.txt  # to change permissions for a file/dir
ls -l test.txt      # to check permissions for file

chmod 400 test.txt  # to set read only permission to only owner
ls -l test.txt      # to check permission of file


# 12 . to change the owner of file
\#echo "test" > own.txt     # create file own.txt with owner & group
'root'
ls -l own.txt           # to check current owner & group
\#chown cdac:check1 own.txt # to change ownership of file 'test.txt' to
user 'cdac' under group 'check', run with root only
ls -l own.txt


#13. to change only group of file
echo "world" > grp.txt      # to create a file with owner & group 'root'
\#chgrp check1 grp.txt      # to change group of file grp.txt to group
'check1', run with root only
ls -l grp.txt           # to check if group is changed


#14. chgrp vs chown
```

chgrp	chown
used to change the ownership of the file	used to change the ownership of any file/dir
applicable for only group	applicable for both user & group

```
#15. umask
# umask stands for user file creation mask
# the default permission of any file/dir is changed to any specific
formation using umask


#
```

```
mkdir d1
umask   # to check umask value, 0022 , do permission is 777-022=755
rwxr-xr-x

touch t.txt #
ls -l t.txt #



#=================================================================================
==================================

#Shell scripting

# it is a program to write a series of commands to execute
# it gathers input from user and execute a program based on the user
inputs
# we can manipulate files & dirs, process & manipulate text and files
# it can help in system administration tasks such as backup, scheduling
any task
# it is helpful in networking to ping any server, to download any files
# "#!/bin/bash" specifies the interpreter that it has to execute a script
# "#!" is called as shebang
# "$" represents variable

#1.  basic Program
vi check.sh
# check.sh contents
#!/bin/bash
pwd
ls
ps
#run command
bash ./check.sh
chmod +x  check.sh  # to change execute permission
bash ./check.sh     # to run script 'check.sh'


#2. basic program
vi check1.sh
```

```
# check1.sh contents
var=Hello
var1=cdac
echo "$var $var1"
#run command
bash ./check1.sh
chmod +x  check1.sh
bash ./check1.sh

#3.  write a shell script  to find a pattern like "cdac" in a file and
once you get the pattern, redirect it to a new file
# check3.txt contents
#!/bin/bash
grep "cdac" filee.txt > out.txt
# filee.txt contents
hello user
welcome to cdac mumbai
youre sent to mumbai
# run command
bash check3.sh


#4. if ... else
# Syntax :
if [condition]
then
    body...
else
    body...
fi
```

```
#5. if ... elif ... else
# Syntax :
if [condition]
then
    body...
elif [condition]
then
    body...
else
    body...
fi


#6. WAP using if else to check if a number is positive or negative
vi check4.sh
#check4.sh contents
#!/bin/bash
echo "Enter a number : "
read num
if [ $num  > 0 ]
then
    echo "$num is positive"
elif [ $num < 0 ]
then
    echo "$num is negative"
else
    echo "$num is zero"
fi
#run command
bash check4.sh
```

```
# alternate method
vi check4_alt.sh
#check4_alt.sh contents
#!/bin/bash
echo "Enter a number : "
read num
if [ $num -gt  0 ]
then
    echo "$num is positive"
elif [ $num -lt 0 ]
then
    echo "$num is negative"
else
    echo "$num is zero"
fi
# run command
bash check4_alt.sh

#7. WAP to check if you're eligible to vote
vi check5.sh
# check5.sh contents
#!/bin/bash
echo "Enter your age : "
read age
if [ $age -gt 17 ]
then
    echo "Eligible to Vote"
else
    echo  "Not eligible to vote"
fi
#run command
bash check5.sh
```

```
#8. for loop
#Syntax:
for in list
do
    body...
done


#9. WAP to print 1-10 numbers using for loop
vi check6.sh
# check6.sh contents
#!/bin/bash
for i in {1..10}
do
    echo "$i"
done
#run command
bash check6.sh
#alternate method
# check6_alt contents
#!/bin/bash
for ((i=0;i<10;i++))
do
    echo "$i"
done
#run command
bash check6_alt.sh
```

```
#10. WAP to check if entered number is even or odd
# check7.sh contents
#!/bin/bash
echo "Enter a number : "
read num
res=($num % 2)
if [ $res -eq 0 ]
then
    echo "number is even"
else
    echo "number is odd"
fi
#run command
bash check7.sh

#11. while loop
#Syntax :
while [condition];
do
    body...
done
# check8.sh contents
i=0
while [ $i -le 10 ]
do
    echo $i
    i=$((i+1))
done
#run command
bash check8.sh
```

```
#12. WAP to check if any user responds yes, it should ask, if user wants
to continue or not
# check 9.sh contents
#!/bin/bash
echo "Do you want to continue ? "
read resp
if [ $resp == "yes" ]
then
    echo "continuing for yes"
    while [ $resp == "yes" ]
    do
        echo "continuing for yes"
        read resp
        if [ $resp == "yes" ]
        then
            echo "continuing for yes"
        else
            echo "stopping for no"
        fi
    done
else
    echo "stopping for no"
fi


#13. Case statements
case in
    pattern1)
        statement1;;
    pattern2)
        statement2;;
esac
```