

Report

Name: Harshit Parsai

Section: K17TA

Roll Number: 09

Registration Number: 11702448

Question 1 > Write a script which keeps track of the first and last user who used the system every wednesday. The entries should be saved in a file named "Wed.txt" under two columns "User_Name" and "Time_In". When the total number of entries in this file reach 50 find out the user with the maximum number of entries.

Solution ->

Step 1- Create a file and mention the field that is "User_Name" and "Time_In" into this file we can do it using touch command and then editing using **nano** editor to edit the file

```
[harshit@localhost ~]$ nano Wed.txt
```

```
GNU nano 2.3.1      File: Wed.txt      Modified
User_Name Time_In
```

Step2 - We will write a **crontab** process that will run on the wednesday and will keep track of the user who are logging in on wednesday and saves the entries in the **tmpf.txt** file we can do this by checking if **who | wc -l** is greater than or equal to 1 is so then we will save the entries of who to tmpf.txt file and we will also run a script on the end of the wednesday so we can edit a crontab using the command **crontab -e**

```
[harshit@localhost ~]$ nano Wed.txt
[harshit@localhost ~]$ crontab -e
```

```
* * * * Wed if [ $(who | wc -l) -ge 1 ]; then who | sort -k4 >> ~/tmpf.txt; fi
59 23 * * Wed ~/process_data.sh
```

Step 3 - We need to write a **script** that will run on the 11:59PM on the Wednesday and that will Check if the entries in the **Wed.txt** files are greater than the 50 if so than it will display the name of the user who logged into the system maximum number of times and the number of times it

got logged into the system if the entries in the Wed.txt file is less than the 50 then we will take the first and the last entry in the tmpf.txt file which keeps the track on the users that got logged into the system on the wednesday and we will save that entries in the Wed.txt file . We can do so by Counting the entries in the Wed file using the command **wc -l ~/Wed.txt | cut -d " " -f1** then we can check the condition using the **-gt** condition in the if condition .If the entries in the Wed.txt is greater then 50 then we can output the name of the user and number of the times he logged in using command written in the then block if the entries in the file less then the 50 then we can take the first entry using **head -n1 ~/tmpf.txt** and then printing the 1st and the 4th field of the returned value of this command to the wed.txt file using **awk '{print \$1 " " \$4}' >> ~/Wed.txt** since the first and the fourth field is name and the time of the user logged in the system and we can take the last entry using **tail -n1 ~/tmpf.txt** and then printing the 1st and the 4th field of the returned value of this command to the wed.txt file using **awk '{print \$1 " " \$4}' >> ~/Wed.txt** since the first and the fourth field is name and the time of the user logged in the system

```
[harshit@localhost ~]$ nano process_data.sh
[harshit@localhost ~]$
```

```
Applications ▾ Places ▾ Terminal ▾ Wed 10:17 🔊 📄 ▾
harshit@localhost:~
File Edit View Search Terminal Help
#!/bin/bash

limit=50
entries=$(wc -l ~/Wed.txt | cut -d " " -f1)

if [ $entries -gt $limit ]
then
    cat ~/Wed.txt | cut -d " " -f1 | sort | uniq -c | sort -r | head -n1 | awk '{pr
int $2 " user logged into system " $1 " times"}' | wall
    echo -n > ~/Wed.txt
else
    head -n1 ~/tmpf.txt | awk '{print $1 " " $2}' >> ~/Wed.txt
    tail -n1 ~/tmpf.txt | awk '{print $1 " " $2}' >> ~/Wed.txt
    echo -n > ~/tmpf.txt
fi

~
~
~
~
~
~
~
~
~
~
~

"process_data.sh" 14L, 416C 11,1-8 All
harshit@localhost:~ 1 / 4
```

Step 4 - To test if our script is working fine or not we can run our cron job every minute to do this we can do like

```
* * * * * if [ $(who | wc -l) -ge 1 ]; then who | sort -k4 >> ~/tmpf.txt; fi

* * * * * ~/process_data.sh

~

~
```

So after 1 minute we can see that tmpf.txt file is created in the current working directory

```
[harshit@localhost ~]$ crontab -e
crontab: installing new crontab
[harshit@localhost ~]$ crontab -l
* * * * * if [ $(who | wc -l) -ge 1 ]; then who | sort -k4 >> ~/tmpf.txt; fi

* * * * * ~/process_data.sh
[harshit@localhost ~]$ ls
Desktop    Downloads  Pictures    Public      tmpf.txt  Wed.txt
Documents  Music      process_data.sh  Templates  Videos
[harshit@localhost ~]$
```

Step 5- to check if the script is working fine or not we can edit the Wed.txt file and can add 50+ user and the time by our own. And after adding a result will be shown after we run the script as

```
Broadcast message from harshit@localhost.localdomain (Mon Apr  6 03:02:01 2020):

harshit user logged into system 138 times
```

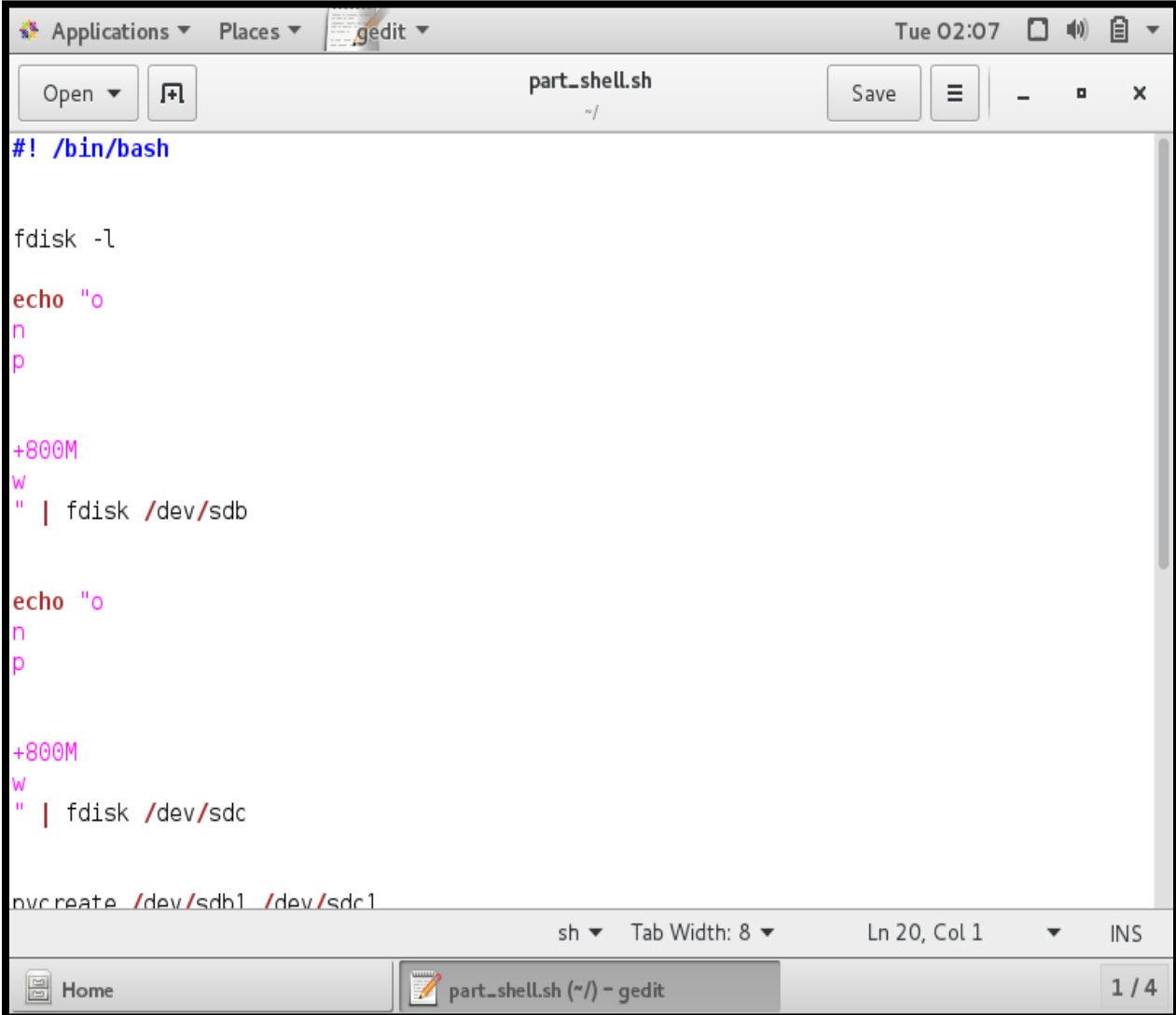
Hence we can see that our script is working fine.

Question 2>For this task, suppose that 2x1GB virtual disks are already present on your system. Write a single bash shell script to create 2x800MB partitions on each disk using fdisk and then bring both partitions into LVM control with the pvcreate command. Create a volume group called vgscript and add both PVs to it. Create three logical volumes each of size 500MB and name them lvscript1, lvscript2, and lvscript3. This question is a continuation of Q2. Write

another bash shell script to create xfs, ext4, and vfat file system structures in each logical volume. Create mount points /mnt/xfs, /mnt/ext4, and /mnt/vfat, and mount the file systems. Include the df command with -h to list the mounted file systems

Solution ->

Step 1 -> Here is the complete bash shell script for the question after that we will show the execution of each and every command with explanation



```
#!/bin/bash

fdisk -l

echo "o
n
p

+800M
w
" | fdisk /dev/sdb

echo "o
n
p

+800M
w
" | fdisk /dev/sdc

pvcreate /dev/sdb1 /dev/sdc1
```

The screenshot shows a gedit editor window titled 'part_shell.sh' with a menu bar (Applications, Places, gedit) and a toolbar (Open, Save, etc.). The script content is as follows:

```
#!/bin/bash

fdisk -l

echo "o
n
p

+800M
w
" | fdisk /dev/sdb

echo "o
n
p

+800M
w
" | fdisk /dev/sdc

pvcreate /dev/sdb1 /dev/sdc1
```

The status bar at the bottom indicates 'sh', 'Tab Width: 8', 'Ln 20, Col 1', and 'INS'. The window title bar shows 'part_shell.sh (~/) - gedit' and '1 / 4'.

```
+800M
w
" | fdisk /dev/sdc

pvcreate /dev/sdb1 /dev/sdc1
vgcreate vgscript /dev/sdb1 /dev/sdc1
lvcreate -n lvscript1 -L 500M vgscript
lvcreate -n lvscript2 -L 500M vgscript
lvcreate -n lvscript3 -L 500M vgscript

mkfs.xfs /dev/vgscript/lvscript1
mkfs.ext4 /dev/vgscript/lvscript2
mkfs.vfat /dev/vgscript/lvscript3

mkdir /mnt/xfs
mkdir /mnt/ext4
mkdir /mnt/vfat

mount /dev/vgscript/lvscript1 /mnt/xfs
mount /dev/vgscript/lvscript2 /mnt/ext4
mount /dev/vgscript/lvscript3 /mnt/vfat
```

So Now here I am explaining what this script actually does by executing the command one - by one

We will **run fdisk -l** to check if there are **two 2*1GB** partition in our system –

```
[harshit@localhost ~]$ sudo fdisk -l
```

After executing this command we can see the virtual disks present in the system and can check if it is good to go for the next task.

```
Disk /dev/sdb: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk /dev/sdc: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

These are the two main disks which we require for this question

Step 2-> We need to create two partitions on these disks of size 800MB which we can do with the command **sudo fdisk /dev/sdb** and providing the required entries

```
[harshit@localhost ~]$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x78827f64.

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): +800M
Partition 1 of type Linux and of size 800 MiB is set

Command (m for help): w
The partition table has been altered!
```

Again by running the command **sudo fdisk /dev/sdc** we can create one more

Step 3 -> we need to bring both the partitions in LVM control so we can do this using the command **pvcreate** as **pvcreate /dev/sdb1 /dev/sdc1** after executing this command we get

```
[harshit@localhost ~]$ sudo pvcreate /dev/sdb1 /dev/sdc1
Physical volume "/dev/sdb1" successfully created.
Physical volume "/dev/sdc1" successfully created.
```

Step 4 -> we need to create a volume group **vgscript** and bring both PVs into it. We can do this using **sudo vgcreate vgscript /dev/sdb1 /dev/sdc1**

```
[harshit@localhost ~]$ sudo vgcreate vgscript /dev/sdb1 /dev/sdc1
Volume group "vgscript" successfully created
```

Step 5 - > We need to create **logical volume lvscript1 , lvscript2 , lvscript3** of size 500MB using this vgscript we can do this by running **lvcreate -n lvscript1 -L 500M vgscript** similarly for lvscript2 and lvscript3

```
[harshit@localhost ~]$ sudo lvcreate -n lvscript1 -L 500M vgscript
Logical volume "lvscript1" created.
[harshit@localhost ~]$ sudo lvcreate -n lvscript2 -L 500M vgscript
Logical volume "lvscript2" created.
[harshit@localhost ~]$ sudo lvcreate -n lvscript3 -L 500M vgscript
Logical volume "lvscript3" created.
[harshit@localhost ~]$
```

Step 5-> We need to create xfs, ext4 , vfat file system in each of this logical volume we can Do so using the command **mkfs.xfs /dev/vgscript/lvscript1 , mkfs.ext4 /dev/vgscript/lvscript2 , mkfs.vfat /dev/vgscript/lvscript3**

```
[harshit@localhost ~]$ sudo mkfs.xfs /dev/vgscript/lvscript1
meta-data=/dev/vgscript/lvscript1 isize=512    agcount=4, agsize=32000 blks
         =                       sectsz=512    attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096    blocks=128000, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096    ascii-ci=0 ftype=1
log      =internal log          bsize=4096    blocks=855, version=2
         =                       sectsz=512    sunit=0 blks, lazy-count=1
realtime =none                  extsz=4096    blocks=0, rtextents=0
[harshit@localhost ~]$
```

```
[harshit@localhost ~]$ sudo mkfs.ext4 /dev/vgscript/lvscript2
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128016 inodes, 512000 blocks
25600 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=34078720
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
[harshit@localhost ~]$ sudo mkfs.vfat /dev/vgscript/lvscript3
mkfs.fat 3.0.20 (12 Jun 2013)
```

Step 6 - We need to create the **mount points /mnt/xfs , /mnt/ext4 , /mnt/vfat** we can do this by making these directories then mounting those file systems in these directories

```
[harshit@localhost mnt]$ sudo mkdir xfs
[harshit@localhost mnt]$ ls
xfs
[harshit@localhost mnt]$ sudo mkdir ext4
[harshit@localhost mnt]$ sudo mkdir vfat
[harshit@localhost mnt]$ sudo mount /dev/vgscript/lvscript1 /mnt/xfs
[harshit@localhost mnt]$ sudo mount /dev/vgscript/lvscript1 /mnt/ext4
[harshit@localhost mnt]$ sudo mount /dev/vgscript/lvscript2 /mnt/ext4
[harshit@localhost mnt]$ sudo mount /dev/vgscript/lvscript3 /mnt/vfat
```

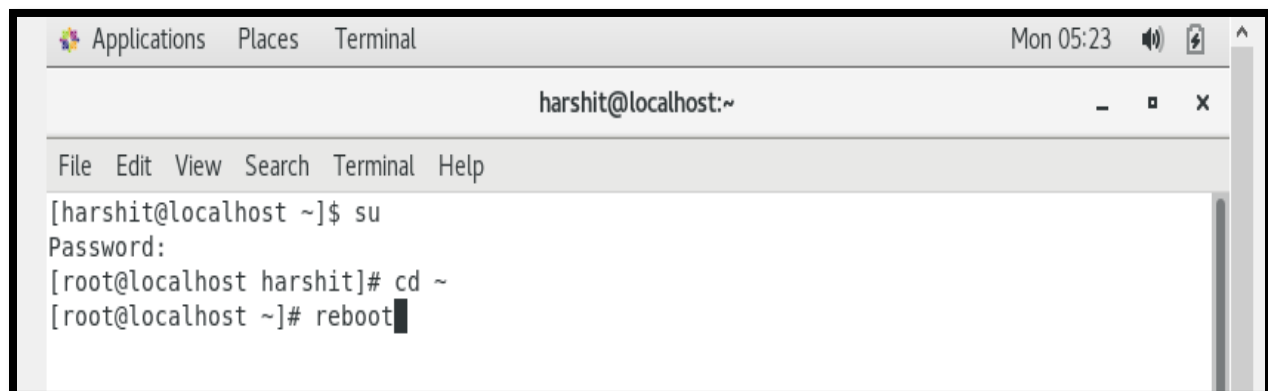
Step 7 - We need to list the mounted file system we can do this by using the command **df -h** after executing this command we get output as


```
[harshit@localhost mnt]$ df -h
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/cl-root        14G       4.0G   9.5G  30% /
devtmpfs                   918M        0   918M   0% /dev
tmpfs                      934M     648K   933M   1% /dev/shm
tmpfs                      934M     8.8M   925M   1% /run
tmpfs                      934M        0   934M   0% /sys/fs/cgroup
/dev/sda1                 1014M     173M   842M  18% /boot
tmpfs                      187M      36K   187M   1% /run/user/1000
/dev/mapper/vgscript-lvscript1 497M      26M   472M   6% /mnt/xfs
/dev/mapper/vgscript-lvscript2 477M     2.3M   445M   1% /mnt/ext4
/dev/mapper/vgscript-lvscript3 500M        0   500M   0% /mnt/vfat
```

Question 3>Reboot your system, and interrupt the countdown in bood-loader menu. Edit the default boot- loader entry, in memory, to abort the process just after the kernel mounts all the file system, but before it hands over control to systemd. At the switch_root prompt, remount the /sysroot file system read/write, then use chroot to go into /sysroot. Change the root password to redhat. Configure the system to automatically perform a full SELinux relabel after boot. Reboot your system again and verify the system is working, by logging in as root at the console.Use redhat as the password.

Solution>

Step1 - To **reboot** our system we need to run reboot command



```

Applications  Places  Terminal  Mon 05:23
harshit@localhost:~
File Edit View Search Terminal Help
[harshit@localhost ~]$ su
Password:
[root@localhost harshit]# cd ~
[root@localhost ~]# reboot

```

Step 2 - To Interrupt the the countdown in boot loader menu we need to **press 'e'**

```
CentOS Linux (3.10.0-1062.18.1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-514.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-1fa454be1035ec4cbe96ed5f5c9cacc4) 7 (Core)
```

Use the ↑ and ↓ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.
The selected entry will be started automatically in 2s.

Step 3 - Edit the default boot loader entry to do that we need to **write rd.break enforcing=0 in place of rhgb quite** this stops the start-up process before the regular filesystem is mounted

```
insmod part_msdos
insmod xfs
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' c395dab2-8\
6dc-45ed-af35-bcd304a8d63b
else
    search --no-floppy --fs-uuid --set=root c395dab2-86dc-45ed-af35-bcd3\
04a8d63b
fi
linux16 /vmlinuz-3.10.0-514.el7.x86_64 root=/dev/mapper/cl-root ro cra\
shkernel=auto rd.lvm.lv=cl/root rd.lvm.lv=cl/swap rd.break enforce=0_LANG=en_U\
S.UTF-8
initrd16 /initramfs-3.10.0-514.el7.x86_64.img
```

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.

Step 4 - Press the **Ctrl+X** then At the **swith-root** then we will **remount the /sysroot** directory in **the rw** then we will **use chroot** as a user to go into the /sysroot directory to change the password of the root the we will type the **passwd** command to change the password the it will ask for the password and then to rewrite the password to change the password

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
```

Step 5 - We exit it from here using the **exit** command 2 times and then we will be taken to the boot loader and then we can use the new password to successfully login to the system as a root

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2# exit
exit
switch_root:/# exit
```

Question 4> Suppose there is a directory named “~/shared/drive” having certain files inside. RECORD every cmd and output typed in the terminal to a file named as acl.txt Add ACL entries to the drive directory (and its contents) that allow members of the gamers group to have read/write access on the files and execute permission on the directory. Restrict the gamer07 user to read access on the files and execute permission on the directory. Restrict the gamer07 user to read access on the files and execute permission on the directory. Write the cmd to review your new ACL settings on /shared/drive. Copy all the ACL permissions of /shared/drive to /shared01/drive01. Delete ACL permissions set for user gamer03 as well as group gamers set on /shared/drive01/file01. WAC to add user jerry read-only and conditional execute permissions to files as well as sub-directories of /shared/dir01 recursively. WAC to add a default access rule for jerry denying any access to /shared/stream directory. Compare the SELinux context of the files present in /shared/drive and /shared01/drive01.

Solution>

Step 1 - This question requires a lot of directories files and users groups. So we will first **create directories files and users and groups** using the command **mkdir , touch , useradd , groupadd** Then we will change the group of the users to gamers group using **usermod** command.

```
[harshit@localhost ~]$ mkdir -p ~/shared/drive
[harshit@localhost ~]$ mkdir -p ~/shared01/drive01
[harshit@localhost ~]$ mkdir -p ~/shared/dir01
[harshit@localhost ~]$ mkdir -p ~/shared/stream
[harshit@localhost ~]$ ls
Desktop    Downloads  part_shell.sh  Public  shared01  Videos
Documents  Music      Pictures       shared  Templates
[harshit@localhost ~]$ cd ~/shared/drive
[harshit@localhost drive]$ touch fill{1..5}
[harshit@localhost drive]$ cd ~/shared/dir01
[harshit@localhost dir01]$ touch file{1..5}
[harshit@localhost dir01]$ cd ~/shared01/drive01
[harshit@localhost drive01]$ touch file0{1..2}
[harshit@localhost drive01]$ ls
file01  file02
```

```
[harshit@localhost ~]$ sudo groupadd gamers
[sudo] password for harshit:
[harshit@localhost ~]$ sudo useradd gamer01
[harshit@localhost ~]$ sudo useradd gamer02
[harshit@localhost ~]$ sudo useradd gamer03
[harshit@localhost ~]$ sudo useradd gamer04
[harshit@localhost ~]$ sudo useradd gamer05
[harshit@localhost ~]$ sudo useradd gamer06
[harshit@localhost ~]$ sudo useradd gamer07
[harshit@localhost ~]$ sudo useradd jerry
```

```
[harshit@localhost ~]$ sudo usermod -aG gamers gamer01
[harshit@localhost ~]$ sudo usermod -aG gamers gamer02
[harshit@localhost ~]$ sudo usermod -aG gamers gamer03
[harshit@localhost ~]$ sudo usermod -aG gamers gamer04
[harshit@localhost ~]$ sudo usermod -aG gamers gamer05
[harshit@localhost ~]$ sudo usermod -aG gamers gamer06
[harshit@localhost ~]$ sudo usermod -aG gamers gamer07
[harshit@localhost ~]$ sudo usermod -aG gamers jerry
```

Step 2 - > We need to have a file which will keep the track of the commands which we are executing and the name of the file is **acl.txt** so to do this we can **start a script** of name **act.txt** using the command **script acl.txt**

```
[harshit@localhost ~]$ script acl.txt
Script started, file is acl.txt
```

Step 3 -> We have to give the users of **gamers group** a read and write permission on the files and the execute permission on the directories of the drive directories so to do this we can use **setfacl** command and can specify the required things to set the permission also we need to give **restrict the gamer07** user of the gamers group to have only read permission on the files and the execute permission on the directories so we can use the same command setfacl and specify the name of the user and the permission we want to give

```
[harshit@localhost ~]$ setfacl -Rm g:gamers:rwX ~/shared/drive
[harshit@localhost ~]$ setfacl -Rm u:gamer07:rX ~/shared/drive
[harshit@localhost ~]$ getfacl ~/shared/drive
getfacl: Removing leading '/' from absolute path names
# file: home/harshit/shared/drive
# owner: harshit
# group: harshit
user::rwx
user:gamer07:r-x
group::rwx
group:gamers:rwx
mask::rwx
other::r-x
```

Step 4 -> Now our next task is to **copy** the permission of the **~/shared/drive** to the **~/shared01/drive01** so we can do this by **getfacl** permission of the **~/shared/drive** and the copying it the **~/shared01/drive01** using the command **setfacl** we can use **--set-file** option of this command to achieve it as-

```
[harshit@localhost ~]$ getfacl ~/shared/drive | setfacl --set-file=- ~/shared01/drive01
getfacl: Removing leading '/' from absolute path names
[harshit@localhost ~]$ getfacl ~/shared01/drive01
getfacl: Removing leading '/' from absolute path names
# file: home/harshit/shared01/drive01
# owner: harshit
# group: harshit
user::rwx
user:gamer07:r-x
group::rwx
group:gamers:rwx
mask::rwx
other::r-x
```

Step 5 -> We need to **remove** the permission of the user **gamer03** and the group **gamers** on **the ~/shared/drive01/file01** we can do this with the command **setfacl** and **giving -x** as the option and mention user name and the group name and the location address on which the permission need to be removed.

```
[harshit@localhost ~]$ setfacl -x u:gamer03 ~/shared01/drive01/file01
```

```
[harshit@localhost ~]$ setfacl -x g:gamers ~/shared01/drive01/file01
```

Step 6 -> We need to write the command to add user **jerry** read only permission to files as well as sub-directories of **/shared/dir01** recursively we can do this with the help of **setfacl** command again and specifying the **d** option in the command to give the default permission and giving no permission on **~/shared/stream** directory

```
[harshit@localhost ~]$ setfacl -Rm u:jerry:rX ~/shared/dir01
[harshit@localhost ~]$ setfacl -d -m u:jerry:- ~/shared/stream
```

Step 7-> here the task is to compare the SELinux context of the file present in the **~/shared/drive** and **~/shared01/drive01** so we can do it with the command **ls -Z**

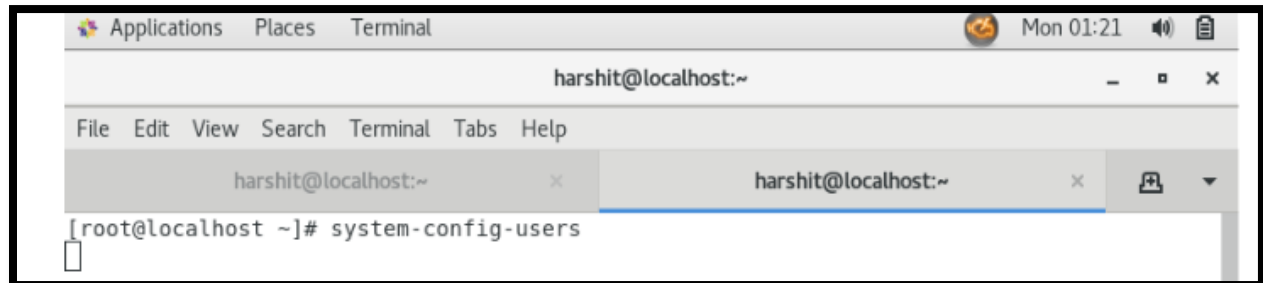
```
[harshit@localhost ~]$ cd ~/shared/drive
[harshit@localhost drive]$ ls -Z
-rw-rw-r--+ harshit harshit unconfined_u:object_r:user_home_t:s0 fill1
-rw-rw-r--+ harshit harshit unconfined_u:object_r:user_home_t:s0 fill2
-rw-rw-r--+ harshit harshit unconfined_u:object_r:user_home_t:s0 fill3
-rw-rw-r--+ harshit harshit unconfined_u:object_r:user_home_t:s0 fill4
-rw-rw-r--+ harshit harshit unconfined_u:object_r:user_home_t:s0 fill5
[harshit@localhost drive]$
[harshit@localhost drive]$ cd ~/shared01/drive01
[harshit@localhost drive01]$ ls -Z
-rw-rw-r--. harshit harshit unconfined_u:object_r:user_home_t:s0 file01
-rw-rw-r--. harshit harshit unconfined_u:object_r:user_home_t:s0 file02
[harshit@localhost drive01]$
```

Question 5 > Open two terminal sessions on **host1** as **root**. Run the **system-config-users** command on one of the terminals. Run a command on the other terminal to determine the **PID** and the **nice** value of the **system-config-users** command. Stop **system-config-users** on the first terminal and re-run it at a lower priority of **+8**. Confirm the new **nice** value of the process by running the appropriate command on the second terminal. Execute the **renice** command on the

second terminal and increase the priority of the system-config-users process to -10, and validate.

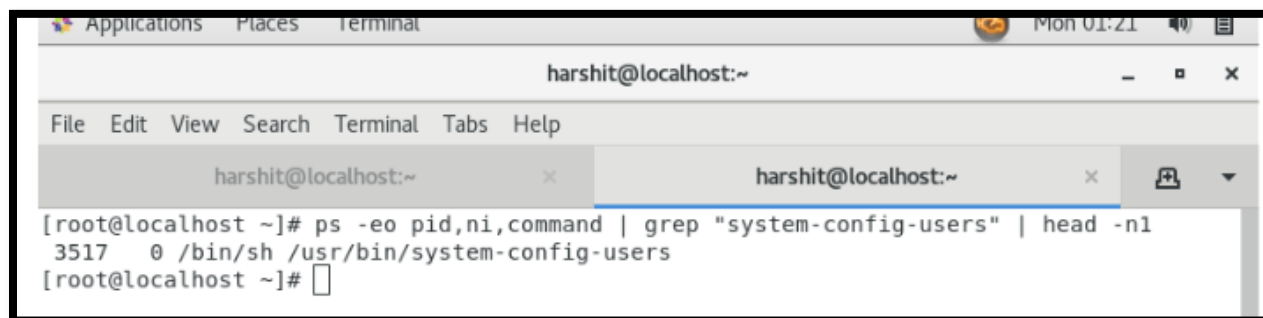
Solution ->

Step 1 - Open the 1st terminal and login as **root** , Then run **system-config-users** command.



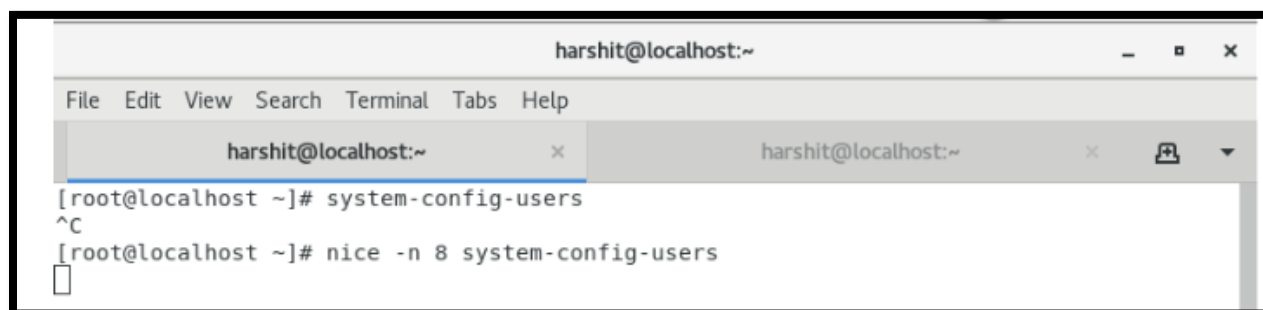
```
Applications  Places  Terminal  Mon 01:21
harshit@localhost:~
File Edit View Search Terminal Tabs Help
harshit@localhost:~ x
harshit@localhost:~ x
[root@localhost ~]# system-config-users
```

Step 2 - We will go to the second terminal and run a command to display the the **process id** of the command we run in the 1st terminal and **the nice value** of the process we can get the process id and nice value using **ps** command.



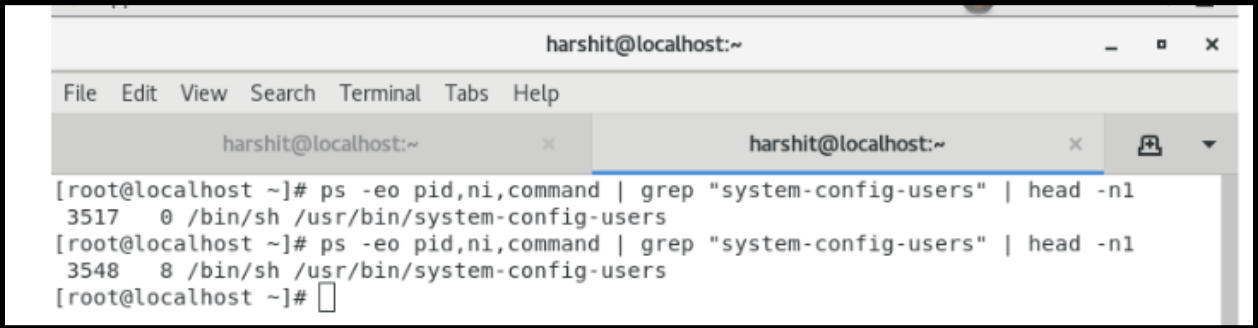
```
Applications  Places  Terminal  Mon 01:21
harshit@localhost:~
File Edit View Search Terminal Tabs Help
harshit@localhost:~ x
harshit@localhost:~ x
[root@localhost ~]# ps -eo pid,ni,command | grep "system-config-users" | head -n1
3517  0 /bin/sh /usr/bin/system-config-users
[root@localhost ~]#
```

Step 3 - Our next task is to stop the command and we can do it using the command **Ctrl + C**, Then we can rerun the process with lower **priority of 8** using the **command nice** and using **-n** argument in it.



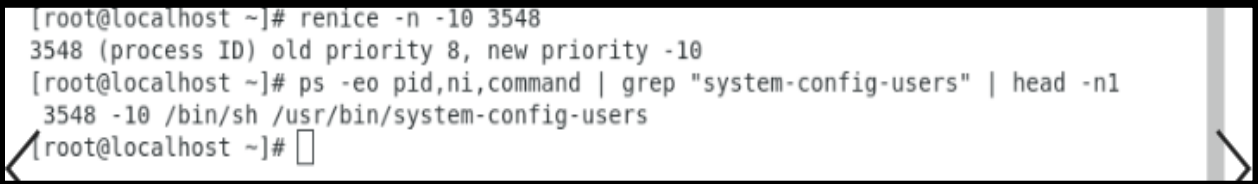
```
harshit@localhost:~
File Edit View Search Terminal Tabs Help
harshit@localhost:~ x
harshit@localhost:~ x
[root@localhost ~]# system-config-users
^C
[root@localhost ~]# nice -n 8 system-config-users
```

Step 4 - We can confirm the nice value of the process in the other terminal using **the ps** command



```
harshit@localhost:~  
File Edit View Search Terminal Tabs Help  
harshit@localhost:~ × harshit@localhost:~ ×  
[root@localhost ~]# ps -eo pid,ni,command | grep "system-config-users" | head -n1  
3517 0 /bin/sh /usr/bin/system-config-users  
[root@localhost ~]# ps -eo pid,ni,command | grep "system-config-users" | head -n1  
3548 8 /bin/sh /usr/bin/system-config-users  
[root@localhost ~]#
```

Step 5- Our next task is to **increase the nice** value of the process to **-10** and we can do it by running **the renice command** in the terminal and providing the new nice value and the process id Of the process. And we can confirm the changes by again running the command **ps** in the same terminal



```
[root@localhost ~]# renice -n -10 3548  
3548 (process ID) old priority 8, new priority -10  
[root@localhost ~]# ps -eo pid,ni,command | grep "system-config-users" | head -n1  
3548 -10 /bin/sh /usr/bin/system-config-users  
[root@localhost ~]#
```

THANKYOU