LAB 1 REPORT

CMSE 353
Security of Software Systems

Team members:

Roa'a Rami Abdel Ra'uof Alqaisi (Team Leader)	19700652
Dalal Totah	19700331
Mohammad Murra	19700717
Khawlah Al-Shubati	19701557

GROUP NO : 01

Computer Engineering Department Eastern Mediterranean University

CMSE 353 Lab 1

Example 1:

States the permissions for each user. The first permission is of the owner and they have the permission read, write, execute and read. The second permission is for the group. And they can execute and read. Others can execute only. The number 142024 is the file size in byte and the date is for last modified and /bin/ls is the file path.

```
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /bin/ls
-rwxr-xr-x. 1 root root 142024 Mar 26  2021 /bin/ls
[RoaaAlqaisi@fedoralinux liveuser]$
```

Example 2:

```
[RoaaAlqaisi@fedoralinux liveuser]$ ls -i /bin/ls
594 /bin/ls
[RoaaAlqaisi@fedoralinux liveuser]$
```

The above command lists the i-node for the /bin/ls file, which is 594

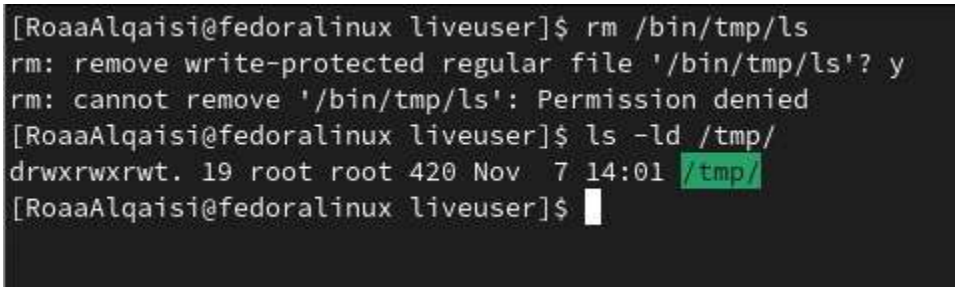
Example 3:

```
[RoaaAlqaisi@fedoralinux liveuser]$ sudo ln /bin/ls /bin/tmp/ls
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /bin/tmp/ls
-rwxr-xr-x. 2 root root 142024 Mar 26  2021 /bin/tmp/ls
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /bin/ls
-rwxr-xr-x. 2 root root 142024 Mar 26  2021 /bin/ls
[RoaaAlqaisi@fedoralinux liveuser]$ ls -i /bin/ls
594 /bin/ls
[RoaaAlqaisi@fedoralinux liveuser]$ ls -i /bin/ls
594 /bin/ls
[RoaaAlqaisi@fedoralinux liveuser]$ ls -i /bin/tmp/ls
594 /bin/tmp/ls
[RoaaAlqaisi@fedoralinux liveuser]$
```

After we created a hard link between the /bin/ and /bin/tmp/ files, the i-nodes for both of these files is the same which is 594. We see that both of the files have 2 as the number of

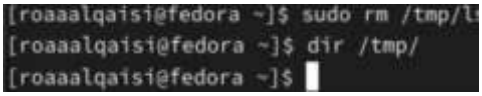
hard links. So any change in one file will affect the other file since they have the same Inode number.

Example 4:



Here the t is the sticky bit which is set for /tmp/ directory so we need to delete that link.

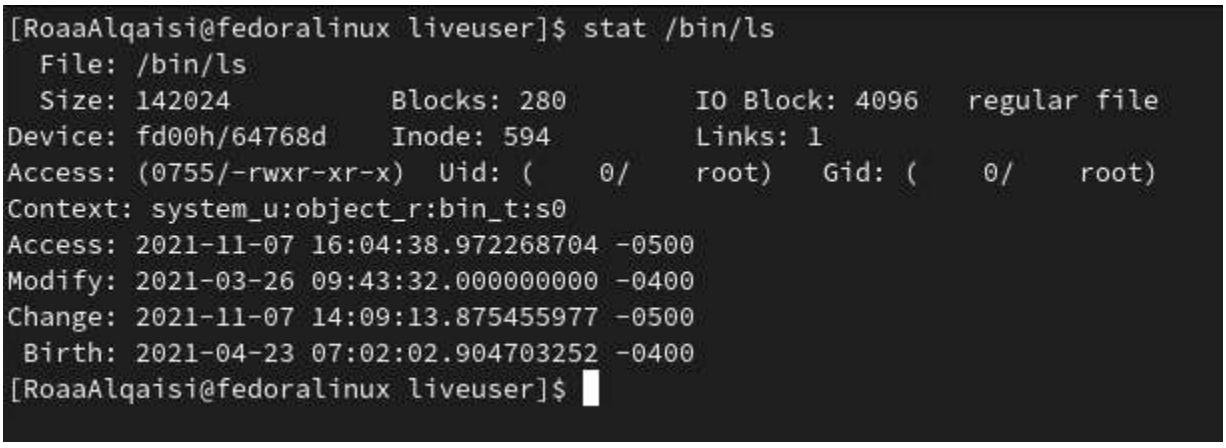
Example 5:



we can delete the link as root using sudo rm /bin/tmp/ls

no content in /bin/tmp/ after /bin/tmp/ls removal.

Example 6:



Here stat command displays more information (statistics)

Uid is for file's owner and Gid is for Files' group and 0775/… are the unix file permissions so here user can read write and execute, group and other users can only read and execute. Stat also lists the dates and times for file access and modification and changes.

Example 7:

```
[RoaaAlqaisi@fedoralinux liveuser]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
systemd-oom:x:998:996:systemd Userspace OOM Killer:/:/sbin/nologin
systemd-timesync:x:997:995:systemd Time Synchronization:/:/sbin/nologin
tss:x:59:59:Account used for TPM access:/dev/null:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:996:994:User for polkitd:/:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
unbound:x:995:992:Unbound DNS resolver:/etc/unbound:/sbin/nologin
dnsmasq:x:994:991:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/sbin/nologin
nm-openconnect:x:993:989:NetworkManager user for OpenConnect:/:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
gluster:x:992:988:GlusterFS daemons:/run/gluster:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
pipewire:x:991:987:PipeWire System Daemon:/var/run/pipewire:/sbin/nologin
geoclue:x:990:986:User for geoclue:/var/lib/geoclue:/sbin/nologin
```

```
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /etc/passwd
-rw-r--r--. 1 root root 2718 Nov  7 13:34 /etc/passwd
[RoaaAlqaisi@fedoralinux liveuser]$
```

We used the file manager “cat” to view all the users in the system using the /etc/passwd command. Then we used the ls –l to list permissions for /etc/passwd

Example 8:

```
[RoaaAlqaisi@fedoralinux liveuser]$ sudo cat /etc/sudoers
[sudo] password for RoaaAlqaisi:
## Sudoers allows particular users to run various commands as
## the root user, without needing the root password.
##
## Examples are provided at the bottom of the file for collections
## of related commands, which can then be delegated out to particular
## users or groups.
##
## This file must be edited with the 'visudo' command.

## Host Aliases
## Groups of machines. You may prefer to use hostnames (perhaps using
## wildcards for entire domains) or IP addresses instead.
# Host_Alias     FILESERVERS = fs1, fs2
# Host_Alias     MAILSERVERS = smtp, smtp2

## User Aliases
## These aren't often necessary, as you can use regular groups
## (ie, from files, LDAP, NIS, etc) in this file - just use %groupname
## rather than USERALIAS
# User_Alias ADMINS = jsmith, mikem


## Command Aliases
## These are groups of related commands...

## Networking
# Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient,
usr/bin/net, /sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sb
n/mii-tool

## Installation and management of software
```

This command lists the users who have the permission to use sudo(from sudo group). They have the same permission as root.

Example 9:

```
[RoaaAlqaisi@fedoralinux liveuser]$ cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:liveuser,RoaaAlqaisi
cdrom:x:11:
mail:x:12:
man:x:15:
dialout:x:18:
floppy:x:19:
games:x:20:
tape:x:33:
video:x:39:
ftp:x:50:
lock:x:54:
audio:x:63:
users:x:100:
nobody:x:65534:
apache:x:48:
utmp:x:22:
utempter:x:35:
systemd-network:x:192:
input:x:999:
kvm:x:36:qemu
render:x:998:
systemd-journal:x:190:
systemd-coredump:x:997:
```

The above command shows the groups on a system and the members enrolled in these groups.

Example 10:

```
mohammad@mohammad-VirtualBox:~/Desktop$ sudo adduser student
[sudo] password for mohammad:
Adding user `student' ...
Adding new group `student' (1003) ...
Adding new user `student' (1003) with group `student' ...
Creating home directory `/home/student' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for student
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
mohammad@mohammad-VirtualBox:~/Desktop$
```

A new user named student has been created

Example 11:

```
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/
/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sb
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/n
cups-pk-helper:x:113:120:user for cups-pk-helper service
r:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/spe
se
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daem
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/u
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/
n/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var
ologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr
gnome-initial-setup:x:124:65534::/run/gnome-initial-setu
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/f
sssd:x:126:131:SSSD system user,,,:/var/lib/sss:/usr/sbi
mohammad:x:1000:1000:Mohammad Murra,,,:/home/mohammad:/b
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sb
newuser555:x:1001:1001:,,,:/home/newuser555:/bin/bash
diwan:x:1002:1002:,,,:/home/diwan:/bin/bash
student:x:1003:1003:,,,:/home/student:/bin/bash
mohammad@mohammad-VirtualBox:~/Desktop$
```

The user “student” has been added to the system containing the users using the cat /etc/passwd command

Example 12:

```
[RoaaAlqaisi@fedoralinux liveuser]$ cat ? ~/mysecret
cat: '?': Permission denied
cat: /home/RoaaAlqaisi/mysecret: No such file or directory
[RoaaAlqaisi@fedoralinux liveuser]$ cat > ~/mysecret
Here is my secret
this is the secret
[RoaaAlqaisi@fedoralinux liveuser]$
```

We created a new file in home directory called “mysecret”.

Example 13:

```
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l ~/mysecret
-rw-rw-r--. 1 RoaaAlqaisi RoaaAlqaisi 38 Nov  7 14:54 /home/RoaaAlqaisi/mysecret
[RoaaAlqaisi@fedoralinux liveuser]$
```

List the information about this file. The owner and group have the right to read and write. Others have the right to read only.

Example 14:

```
[RoaaAlqaisi@fedoralinux liveuser]$ sudo mkdir /home/tmp
[RoaaAlqaisi@fedoralinux liveuser]$ touch /home/tmp/somefile
touch: cannot touch '/home/tmp/somefile': Permission denied
[RoaaAlqaisi@fedoralinux liveuser]$ sudo touch /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /home/tmp/somefile
-rw-r--r--. 1 root root 0 Nov  7 14:57 /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$ chmod 770 /home/tmp/somefile
chmod: changing permissions of '/home/tmp/somefile': Operation not permitted
[RoaaAlqaisi@fedoralinux liveuser]$ sudo chmod 770 /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /home/tmp/somefile
-rwxrwx---. 1 root root 0 Nov  7 14:57 /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$
```

The chmod allows us to change the privileges on certain file. Using chmod 770, we added the execute property to the owner and group, and removed any privilege for others. So the permissions have changed from 644 to 770.

Example 15:

```
[RoaaAlqaisi@fedoralinux liveuser]$ sudo chmod u-x /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /home/tmp/somefile
-rw-rwx---. 1 root root 0 Nov  7 14:57 /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$
```

Another way to change the privileges is by using the letters r,w,x using the arithmetics “+” and “-” to add or remove privileges. In this example we removed the execute property from the owner.

Example 16:

```
[RoaaAlqaisi@fedoralinux liveuser]$ sudo chmod o+x /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l /home/tmp/somefile
-rw-rwx--x. 1 root root 0 Nov  7 14:57 /home/tmp/somefile
[RoaaAlqaisi@fedoralinux liveuser]$
```

We added the privilege of execution to the others on the file “somefile”

Example 17:

```
[RoaaAlqaisi@fedoralinux liveuser]$ chmod 660 ~/mysecret
[RoaaAlqaisi@fedoralinux liveuser]$ ls -l ~/mysecret
-rw-rw----. 1 RoaaAlqaisi RoaaAlqaisi 38 Nov  7 14:54 /home/RoaaAlqaisi/mysecret
[RoaaAlqaisi@fedoralinux liveuser]$
```

In this example we used the chmod to give the user and group the right to write and read and eliminate any others privileges.

Example 18:

```
[RoaaAlqaisi@fedoralinux liveuser]$ su student
Password:
[student@fedoralinux liveuser]$ cat /home/RoaaAlqaisi/mysecret
cat: /home/RoaaAlqaisi/mysecret: Permission denied
[student@fedoralinux liveuser]$
```

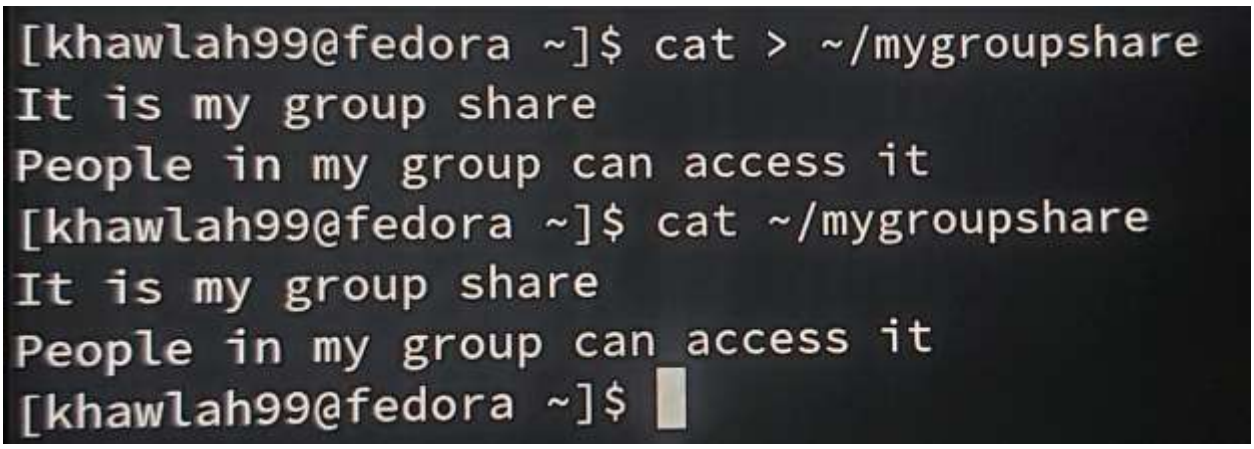
Thus when we try to access the “mysecret” file using the user student, the permission is denied.

Example 19:

```
[khawlah99@fedora ~]$ touch ~/myshare
[khawlah99@fedora ~]$ chmod 666 ~/myshare
[khawlah99@fedora ~]$ ls -l ~/myshare
-rw-rw-rw-. 1 khawlah99 khawlah99 39 Nov  7 23:37 /home/khawlah99/myshare
[khawlah99@fedora ~]$ kwrite ~/myshare
QSocketNotifier: Can only be used with threads started with QThread
"/proc/5686/root"
"/proc/5686/root"
[khawlah99@fedora ~]$ cat ~/myshare
It is to be shared with all the people
[khawlah99@fedora ~]$ sudo chmod 701 /home/khawlah99
[khawlah99@fedora ~]$ su student
Password:
[student@fedora khawlah99]$ cat /home/khawlah99/myshare
It is to be shared with all the people
[student@fedora khawlah99]$
```

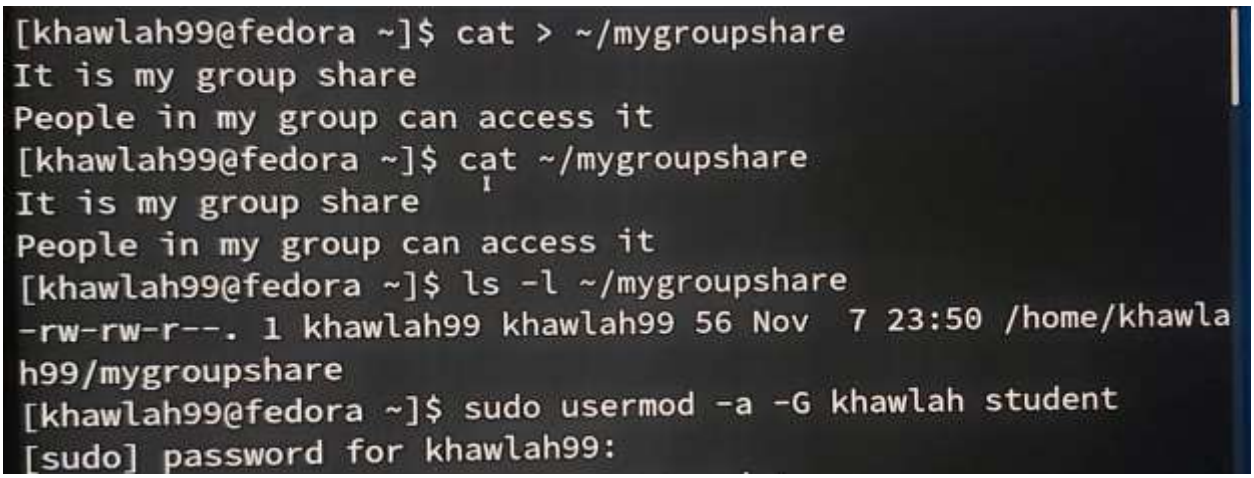
Creating the “myshare” file using touch keyword and allowing users, groups, and others to have read and write permissions. We wrote to that file using the first user then we edited the properties. After that, we opened it again using the student user and the same information is displayed

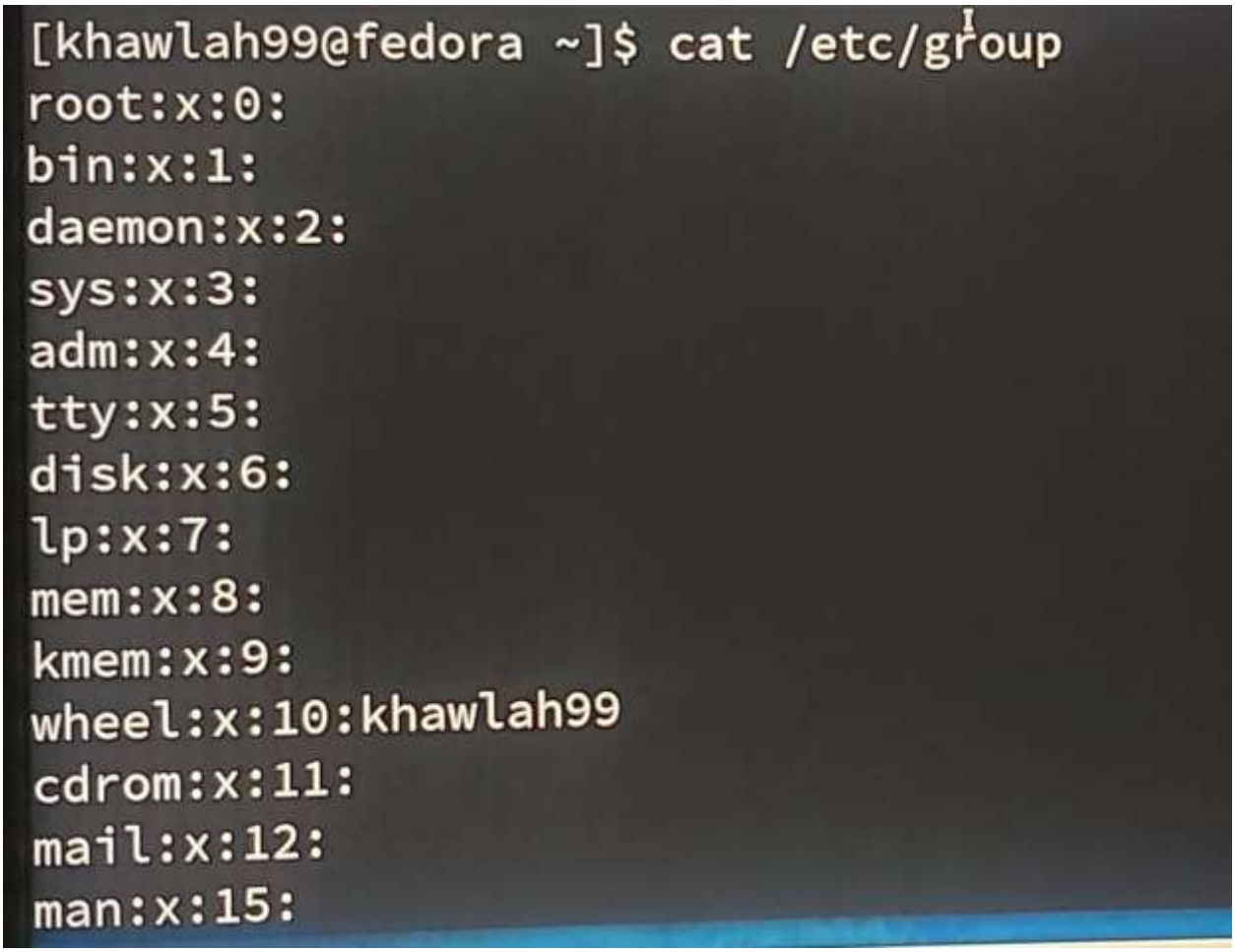
Example 20:

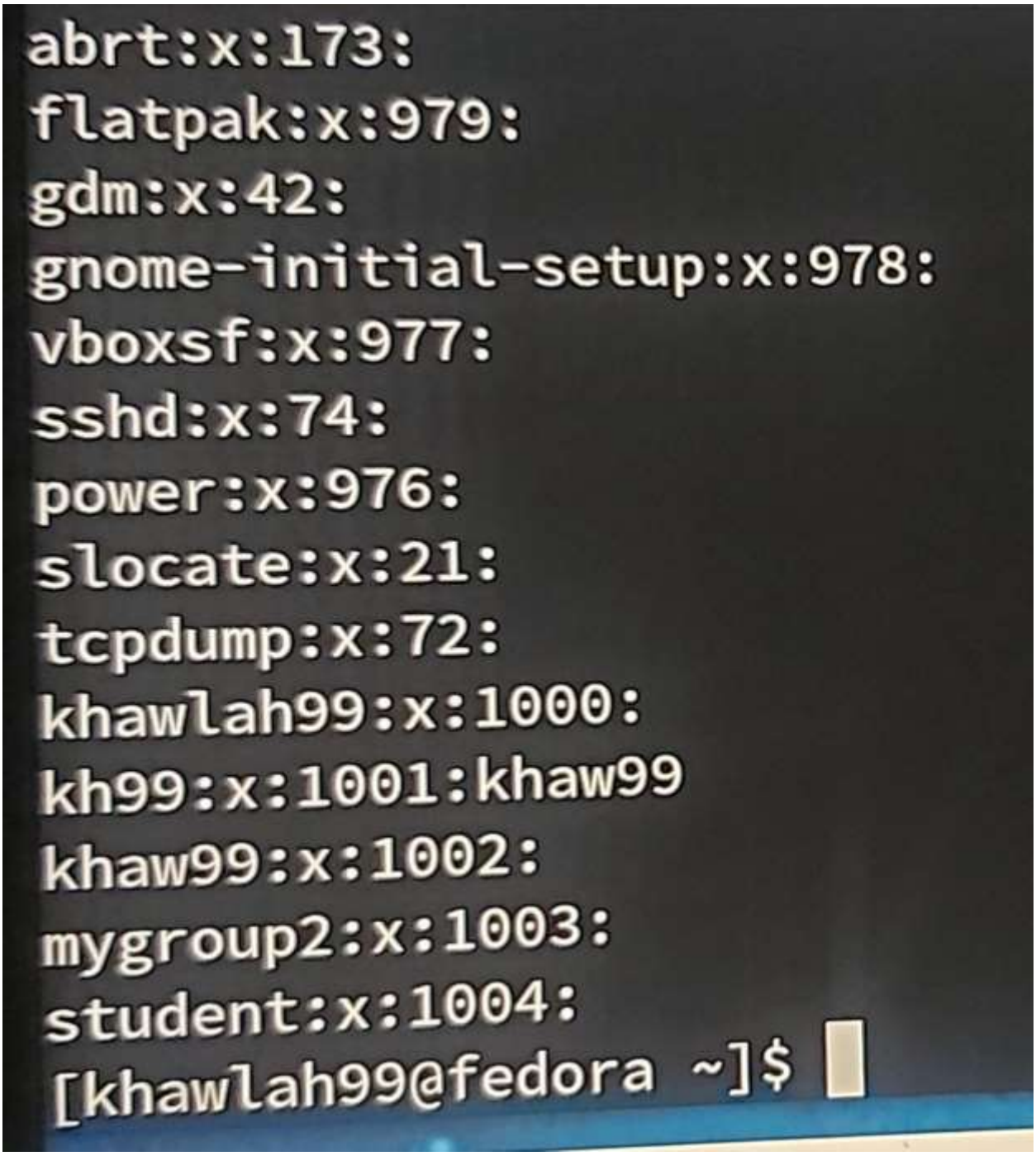


We created the file “mygroupshare”. Only read access is granted to everyone in the group.

Example 21:

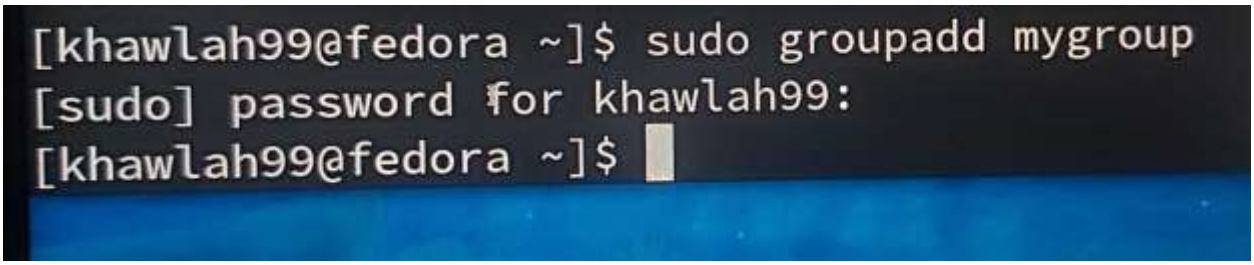



[khawlah99@fedora ~]$ cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:khawlah99
cdrom:x:11:
mail:x:12:
man:x:15:



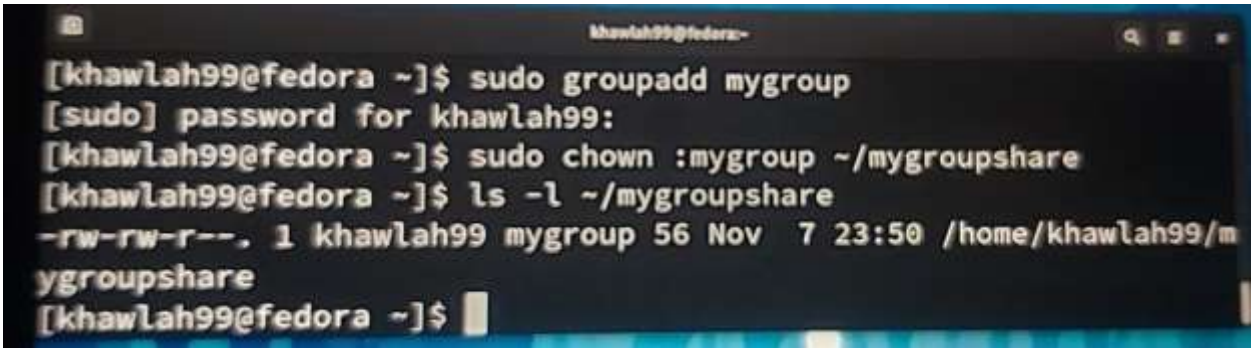
Here we added a user student to group khawlah99 by sudo usermod -a -G then we listed groups to check if it was added.

Example 22:



Here we created a new group called mygroup using groupadd

Example 23:



Here we change the owner group of mygroubshare from khawlah99 to mygroup

Example 24:

Here we changed back mygroupshare to khawlah99 then gave it group only read access to mygroupshare. Then checked the permissions given.

Example 25:

```
[roaaalqaisi@fedora ~]$ dir
Desktop    Downloads  mygroupshare  Pictures  Templates
Documents  Music      myshare       Public    Videos
[roaaalqaisi@fedora ~]$ mkdir tmp
[roaaalqaisi@fedora ~]$ dir
Desktop    Downloads  mygroupshare  Pictures  Templates  Videos
Documents  Music      myshare       Public    tmp
[roaaalqaisi@fedora ~]$ dir tmp
[roaaalqaisi@fedora ~]$ touch tmp/test1 tmp/test2 tmp/test3
[roaaalqaisi@fedora ~]$ dir tmp
test1  test2  test3
[roaaalqaisi@fedora ~]$
```

We list directories and made a new directory called tmp. Then we checked if it was created after that we added three new files test1, test2, test3 then made sure these files were created in tmp directory.

Challenge 1:

```
mohammad@mohammad-VirtualBox:~/Desktop$ mkdir test
mohammad@mohammad-VirtualBox:~/Desktop$ touch test/test1 test/test2 test/test3
mohammad@mohammad-VirtualBox:~/Desktop$ ls -l test
total 0
-rw-rw-r-- 1 mohammad mohammad 0 Noε   7 23:19 test1
-rw-rw-r-- 1 mohammad mohammad 0 Noε   7 23:19 test2
-rw-rw-r-- 1 mohammad mohammad 0 Noε   7 23:19 test3
mohammad@mohammad-VirtualBox:~/Desktop$ chmod -R 755 test
mohammad@mohammad-VirtualBox:~/Desktop$ ls -l test
total 0
-rwxr-xr-x 1 mohammad mohammad 0 Noε   7 23:19 test1
-rwxr-xr-x 1 mohammad mohammad 0 Noε   7 23:19 test2
-rwxr-xr-x 1 mohammad mohammad 0 Noε   7 23:19 test3
mohammad@mohammad-VirtualBox:~/Desktop$
```

We created the folder test followed by three files inside which are test1, test2, test3. The files information have been updated recursively using the chmod –R and allowing the user to read, write, and execute while limiting the group and other users to read and execute only.

Example 26:

```
[roaaalqaisi@fedora ~]$ umask -S
u=rwx,g=rwx,o=rx
[roaaalqaisi@fedora ~]$ umask -p
umask 0002
[roaaalqaisi@fedora ~]$ umask
0002
[roaaalqaisi@fedora ~]$
```

Using the umask to check the default option. On this machine, the newly created files will have “775” permissions since the umask is set to 002. Another way to check it is using the umask –S which shows the permissions by letters(rwx-rwx-rx).

Challenge 2:

```
mohammad@mohammad-VirtualBox:~/Desktop$ touch test/test4
mohammad@mohammad-VirtualBox:~/Desktop$ ls -l test/test4
-rw-rw-r-- 1 mohammad mohammad 0 Νοε   8 11:52 test/test4
mohammad@mohammad-VirtualBox:~/Desktop$ umask 066
mohammad@mohammad-VirtualBox:~/Desktop$ touch test/test5
mohammad@mohammad-VirtualBox:~/Desktop$ ls -l test/test5
-rw------- 1 mohammad mohammad 0 Νοε   8 11:53 test/test5
mohammad@mohammad-VirtualBox:~/Desktop$
```

The test 4 file has the permissions rw-rw-r. After changing the umask 066, only the user is now able to read and write to the file while the rest have no permission to read and write.

Example 27:

```
[roaaalqaisi@fedora ~]$ umask -S
u=rwx,g=rwx,o=rx
[roaaalqaisi@fedora ~]$ umask -p
umask 0002
[roaaalqaisi@fedora ~]$ umask
0002
[roaaalqaisi@fedora ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 32552 Jul 23 03:33 /usr/bin/passwd
[roaaalqaisi@fedora ~]$
```

The owner running this file is the root since we are using the passwd, so the letter s appears to indicate that the UID is running as effective UID

Example 28:

```
[bob@fedora roaaalqaisi]$ stat /usr/bin/passwd
  File: /usr/bin/passwd
  Size: 32552           Blocks: 64         IO Block: 4096   regular file
Device: 20h/32d Inode: 3718        Links: 1
Access: (4755/-rwsr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Context: system_u:object_r:passwd_exec_t:s0
Access: 2021-11-08 13:52:32.585139239 +0300
Modify: 2021-07-23 03:33:08.000000000 +0300
Change: 2021-11-08 13:12:36.714128977 +0300
 Birth: 2021-11-08 13:12:36.709129054 +0300
[bob@fedora roaaalqaisi]$ su bob
Password:
[bob@fedora roaaalqaisi]$ passwd
Changing password for user bob.
Current password:
```

Using stat /usr/bin/passwd to display the file system info then we switched to another user (bob)

Example 29:

```
diwan@mohammad-VirtualBox:/home/mohammad/Desktop$ ps -af
UID          PID    PPID  C STIME TTY          TIME CMD
mohammad     747     745  0 09:07 tty2     00:00:11 /usr/lib/xorg/Xorg vt2 -dis
mohammad     847     745  0 09:07 tty2     00:00:00 /usr/libexec/gnome-session-
root        2457    1350  0 12:44 pts/0    00:00:00 su diwan
diwan       2458    2457  0 12:44 pts/0    00:00:00 bash
root        2470    2458  0 12:46 pts/0    00:00:00 su mohammad
mohammad    2471    2470  0 12:46 pts/0    00:00:00 bash
root        2480    2471  0 12:47 pts/0    00:00:00 su diwan
diwan       2481    2480  0 12:47 pts/0    00:00:00 bash
diwan       2494    2481  0 12:51 pts/0    00:00:00 ps -af
diwan@mohammad-VirtualBox:/home/mohammad/Desktop$
```

Using ps –af, we were able to view all the processes. It runs with root being the UID, although it was launched using a different user(bob)

Example 30:

```
mohammad@mohammad-VirtualBox:~$ stat accessmysecret
  File: accessmysecret
  Size: 17056           Blocks: 40         IO Block: 4096   regular file
Device: 805h/2053d      Inode: 13953       Links: 1
Access: (0775/-rwxrwxr-x)  Uid: ( 1000/mohammad)   Gid: ( 1000/mohammad)
Access: 2021-11-13 22:08:37.332766132 +0200
Modify: 2021-11-13 22:08:22.488733652 +0200
Change: 2021-11-13 22:08:22.488733652 +0200
 Birth: -
mohammad@mohammad-VirtualBox:~$
```

The stats of the file “accessmysecret” has been described with uid being the root.

Example 31:

```
mohammad@mohammad-VirtualBox:~$ chmod 600 mysecret
mohammad@mohammad-VirtualBox:~$ ls -l mysecret
-rw------- 1 mohammad mohammad 22 Νοε   8 12:58 mysecret
mohammad@mohammad-VirtualBox:~$
```

The file “my secret” should be accessible only by the owner, hence we used the chmod command to perform this action (chmod 600)

Example 32:

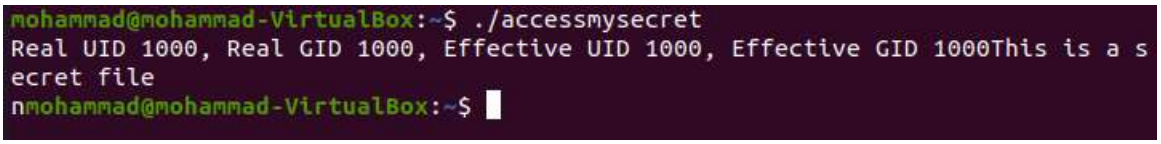
```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>

int main(){
        printf("Real UID %d, Real GID %d, Effective UID %d, Effective GID %d",
getuid(), getgid(), geteuid(), getegid());
        FILE *fp = fopen("mysecret", "r");
        if(fp==NULL)
        {
                printf("Error : could not open file!");
                exit(EXIT_FAILURE);
        }
        char c;
        while((c=getc(fp))!=EOF)
        {
                putchar(c);
        }
        putchar('/n');
        return EXIT_SUCCESS;
}

```

The file accessmysecret now contains the written c code.

Example 33:

We compiled the c code of the accessmysecret file and created and executable file for it and set the permission for the file using chmod u+s.

Example 34:



After the running the program, the output that was displayed is the real end effective identity of the user

Example 35:

After switching to another user, the file (accessmysecret) was executed, although this user does not have direct access on the file itself that was created by the main user.

Challenge 3:

```
mohammad@mohammad-VirtualBox:~$ rm /home/newuser555/tmp7/mysecret
mohammad@mohammad-VirtualBox:~$ rm /home/newuser555/tmp7/access1
mohammad@mohammad-VirtualBox:~$ sudo ln /home/mohammad/accessmysecret /home/new
user555/tmp7/access1
[sudo] password for mohammad:
mohammad@mohammad-VirtualBox:~$ sudo ln /home/mohammad/myshare /home/newuser555
/tmp7/mysecret
mohammad@mohammad-VirtualBox:~$
```

```
newuser555@mohammad-VirtualBox:/home/mohammad$ /home/newuser555/tmp7/access1
Real UID 1001, Real GID 1001, Effective UID 1000, Effective GID 1001 This is a
secret file
 This is a CMSE353 file
```

We tried to gain access to one of the root user's file which is my share using the accessmysecret program code to read the content of the file. With the help of hardlink, the new user (newuser555) was able to read the content of that file

Challenge 4:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>

int main(){
        printf("Real UID %d, Real GID %d, Effective UID %d, Effective GID
 %d",
getuid(), getgid(), geteuid(), getegid());
        FILE *fp = fopen("mysecret", "r");

        int UID;
        int EUID;

        UID=(int)getuid;
        EUID=(int)geteuid;

        if(fp==NULL && UID!=EUID)
        {
                printf("Error : could not open file!");
                exit(EXIT_FAILURE);
        }
        char c;
        while((c=getc(fp))!=EOF)
        {
                putchar(c);
        }
```

```
newuser555@mohammad-VirtualBox:/home/mohammad$ /home/newuser555/tmp7/access1
Real UID 1001, Real GID 1001, Effective UID 1001, Effective GID 1001Error : cou
ld not open file!newuser555@mohammad-VirtualBox:/home/mohammad$
```

In challenge 3, we see that the id of the new user is different from the root user. To solve this vulnerability, we modified the c code of the accessmysecret file to prevent such security problems by displaying an error message if the user is different from the root user.

Challenge 5:

```
int UID;
int EUID;

UID=(int)getuid;
EUID=(int)geteuid;

if(fp==NULL && UID!=EUID)
{
        printf("Error : could not open file!");
        exit(EXIT_FAILURE);
}
char c;
while((c=getc(fp))!=EOF)
{
       if (c=='\n')
              break;
        putchar(c);
}
```

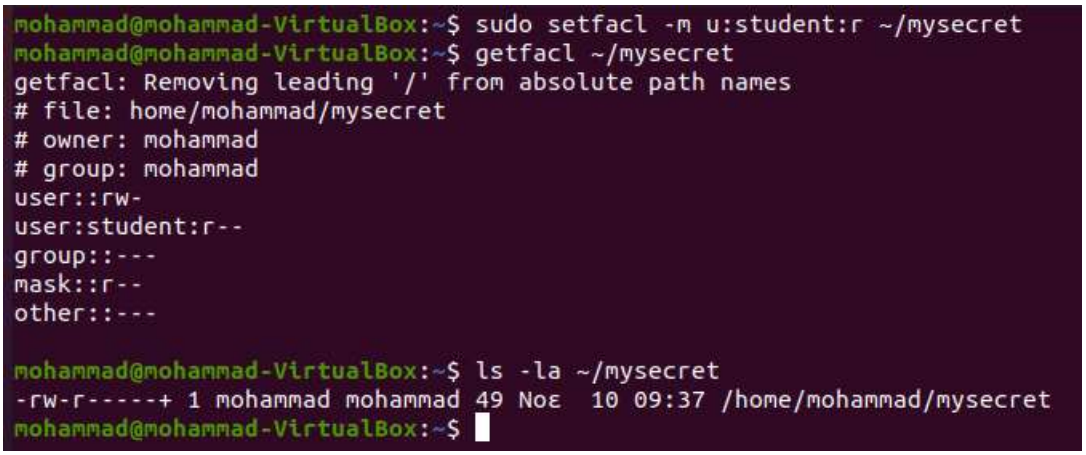
The code has been modified so that when the character reaches \n, indicating a new line, it will break out of the loop and only the first line of the text file will be displayed.

Challenge 6:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>

int main(){
        printf("Real UID %d, Real GID %d, Effective UID %d, Effective GID
 %d",
getuid(), getgid(), geteuid(), getegid());
        FILE *fp = fopen("mysecret", "r");

        int UID;
        int EUID;

        UID=(int)getuid;
        EUID=(int)geteuid;

        if(fp==NULL && UID!=EUID)
        {
                printf("Error : could not open file!");
                exit(EXIT_FAILURE);
        }
        char c;
        while((c=getc(fp))!=EOF)
        {
                putchar(c);
        }
```

The code modification which was done in challenge 4 works perfectly for challenge 6, since the condition to check the user was already there

Example 36:



We set the ACL of the file mysecrets so that the student is granted user read access to it using the setfacl command. The file ACL is detected by displaying a “+” sign when using the ls –la command. The permissions of the file can be also displayed using the getfacl.