

CMSE353 Fall 2021

Access Control in Linux

Contents


CMSE353 Fall 2021	
Access Control in Linux	
4. Unix file permissions and inodes	5
Example 1	6
Example 2	6
Example 3	7
Example 4	7
Example 5	7
5. Octal representation of permissions	7
Example 6	8
6. Changing file permissions on Linux system, adding users, groups, changing file owner and group	8
Example 7	9
Example 8	9
Example 9	10
Example 10	11
Example 11	11
Example 12	12
Example 14	12
Example 15	12
Example 16	13
Example 17	13
Example 18	13
Example 19	13
Example 20	14
Example 21	14
Example 22	14
Example 23	15

Example 24	15
Example 25	15
Challenge 1	15
7. Command umask	15
Example 26	16
Challenge 2	16
8. Set UID (SUID)	16
Example 27	16
Example 28	17
Example 29	17
Example 30	18
9. Writing a SUID program in C	18
Example 31	18
Example 32	19
Example 33	20
Example 34	20
Example 35	21
Challenge 3	21
Challenge 4	21
Challenge 5	21
Challenge 6	21
10. Linux Extended ACLs	22
Example 36	22

The distribution of tasks

Sinem & Emre : 

Seyit : 

Jamshid : 

Example - 1

```
[parallels@fedora ~]$ ls -l /bin/ls
-rwxr-xr-x. 1 root root 138024 Jul  7 19:20 /bin/ls
[parallels@fedora ~]$
```

Defines the location of the file on disk, along with attributes including the Unix file permissions, and access times for the file.

Example - 2

```
[parallels@fedora ~]$ ls -i /bin/ls
197076 /bin/ls
[parallels@fedora ~]$
```

ls-i display the inode number of the given path.

Example – 3

```
[parallels@fedora ~]$ sudo mkdir /bin/tmp
[parallels@fedora ~]$ sudo ln /bin/ls /bin/tmp/ls
[parallels@fedora ~]$ ls -l /bin/tmp/ls
-rwxr-xr-x. 2 root root 138024 Jul  7 19:20 /bin/tmp/ls
[parallels@fedora ~]$ ls -l /bin/ls
-rwxr-xr-x. 2 root root 138024 Jul  7 19:20 /bin/ls
[parallels@fedora ~]$ ls -i /bin/tmp/ls
197076 /bin/tmp/ls
[parallels@fedora ~]$ ls -i /bin/ls
197076 /bin/ls
[parallels@fedora ~]$
```

We see that the both files share the same inode number. if changing one of these files will affect the other.

Example – 4

```
[parallels@fedora ~]$ rm /bin/tmp/ls
rm: remove write-protected regular file '/bin/tmp/ls'? y
rm: cannot remove '/bin/tmp/ls': Permission denied
[parallels@fedora ~]$ ls -ld /tmp/
drwxrwxrwt. 20 root root 400 Nov  6 23:06 /tmp/
[parallels@fedora ~]$
```

We can see the /bin/ls file as a normal user but can not delete that link from sticky bit for /tmp/ directory.

Example – 5

```
[parallels@fedora ~]$ sudo rm /bin/tmp/ls
[parallels@fedora ~]$ ls -l /bin/tmp/ls
ls: cannot access '/bin/tmp/ls': No such file or directory
[parallels@fedora ~]$ ls -l /bin/tmp
total 0
[parallels@fedora ~]$
```

If we add sudo to delete a sticky bit

Example – 6

```
[parallels@fedora ~]$ stat /bin/ls
  File: /bin/ls
  Size: 138024      Blocks: 272      IO Block: 4096   regular file
Device: 24h/36d Inode: 197076      Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (  0/   root)   Gid: (  0/   root)
Context: system_u:object_r:bin_t:s0
Access: 2021-11-06 23:21:18.681124772 +0200
Modify: 2021-07-07 19:20:48.000000000 +0300
Change: 2021-11-06 23:19:44.744081431 +0200
 Birth: 2021-08-02 11:22:38.228359064 +0300
[parallels@fedora ~]$
```

Stat command display the output includes the access rights, along with the last time the file was accessed, modified, and when the inode was last changed

Example – 7

```

[parallels@fedora ~]$ 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
chrony:x:989:984:/var/lib/chrony:/sbin/nologin
saslauth:x:988:76:Saslauthd user:/run/saslauthd:/sbin/nologin
radvd:x:75:75:radvd user:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
qemu:x:107:107:qemu user:/sbin/nologin
openvpn:x:987:982:OpenVPN:/etc/openvpn:/sbin/nologin
nm-openvpn:x:986:981:Default user for running openvpn spawned by NetworkManager:/sbin/nologin
colord:x:985:980:User for colord:/var/lib/colord:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
abrt:x:173:173:/etc/abrt:/sbin/nologin
flatpak:x:984:979:User for flatpak system helper:/sbin/nologin
gdm:x:42:42:/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:983:978:/run/gnome-initial-setup:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/usr/share/empty.sshd:/sbin/nologin
tcpdump:x:72:72:/sbin/nologin
parallels:x:1000:1000:emrecakir:/home/parallels:/bin/bash
[parallels@fedora ~]$ ls -l /etc/passwd
-rw-r--r--. 1 root root 2573 Nov  6 23:27 /etc/passwd
[parallels@fedora ~]$ █

```

We can read every user

Example – 8

```
[parallels@fedora ~]$ sudo cat /etc/sudoers
## 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/lwconfig, /sbin/mii-tool

## Installation and management of software
# Cmnd_Alias   SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum

## Services
# Cmnd_Alias   SERVICES = /sbin/service, /sbin/chkconfig, /usr/bin/systemctl start, /usr/bin/systemctl stop, /usr/bin/systemctl reload, /usr/bin/systemctl restart, /usr/bin/systemctl status, /usr/bin/systemctl enable, /usr/bin/systemctl disable

## Updating the locate database
# Cmnd_Alias   LOCATE = /usr/bin/updatedb

## Storage
# Cmnd_Alias   STORAGE = /sbin/fdisk, /sbin/sfdisk, /sbin/parted, /sbin/partprobe, /bin/mount, /bin/umount

## Delegating permissions
# Cmnd_Alias   DELEGATING = /usr/sbin/visudo, /bin/chown, /bin/chmod, /bin/chgrp

## Processes
# Cmnd_Alias   PROCESSES = /bin/nice, /bin/kill, /usr/bin/kill, /usr/bin/killall

## Drivers
# Cmnd_Alias   DRIVERS = /sbin/modprobe

# Defaults specification
#

# found in the system group database were passed to the group
# plugin, if any. Starting with 1.8.15, only groups of the form
# %:group are resolved via the group plugin by default.
# We enable always_query_group_plugin to restore old behavior.
# Disable this option for new behavior.
Defaults      always_query_group_plugin

Defaults      env_reset
Defaults      env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS"
Defaults      env_keep += "MAIL QDIR USERNAME LANG LC_ADDRESS LC_CTYPE"
Defaults      env_keep += "LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES"
Defaults      env_keep += "LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE"
Defaults      env_keep += "LC_TIME LC_ALL LANGUAGE LANGUAS _XKB_CHARSET XAUTHORITY"

#
# Adding HOME to env_keep may enable a user to run unrestricted
# commands via sudo.
#
# Defaults      env_keep += "HOME"

Defaults      secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/var/lib/snapd/snap/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##     user    MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)    ALL

## Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users    ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users    localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
[parallels@fedora ~]$
```

The users allowed doing sudo (from sudo group) are enlisted file

Example – 9

```
[parallels@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:parallels
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:
systemd-resolve:x:193:
systemd-oom:x:996:
systemd-timesync:x:995:
tss:x:59:
dbus:x:81:
polkitd:x:994:
avahi:x:70:
dip:x:40:
printadmin:x:993:
unbound:x:992:
dnsmasq:x:991:
ssh_keys:x:990:
nm-openconnect:x:989:
usbmuxd:x:113:
gluster:x:988:
rtkit:x:172:
pipewire:x:987:
geoclue:x:986:
brlapi:x:985:
chrony:x:984:
saslauth:x:76:
```

We see that sudo group users have the same permissions as root

Example – 10

```
[parallels@fedora ~]$ sudo adduser student
[sudo] password for parallels:
[parallels@fedora ~]$ sudo passwd student
Changing password for user student.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[parallels@fedora ~]$
```

If we want to create a new user, you can use sudo adduser command

Example – 11

```
parallels:x:1000:1000:Parallels:/home/parallels:/bin/bash
student:x:1001:1001::/home/student:/bin/bash
[parallels@fedora ~]$
```

A new user student is created

Example – 12

```
[parallels@fedora ~]$ cat ? ~/mysecret
cat: '?': No such file or directory
cat: /home/parallels/mysecret: No such file or directory
[parallels@fedora ~]$ cat > ~/mysecret
It is my secret
Here it is
My secret is here
[parallels@fedora ~]$
```

The cat command is used to read the contents of the file. But if the > sign is placed after the cat command, a new file is created.

Example – 13


```
[parallels@fedora ~]$ ls -l ~/mysecret
-rw-rw-r--. 1 parallels parallels 45 Nov  8 20:22 /home/parallels/mysecret
[parallels@fedora ~]$
```

First view the permissions of your newly created file.

Example – 14

```
[parallels@fedora ~]$ mkdir /home/tmp
mkdir: cannot create directory '/home/tmp': Permission denied
[parallels@fedora ~]$ sudo mkdir /home/tmp
[sudo] password for parallels:
[parallels@fedora ~]$ touch /home/tmp/somefile
touch: cannot touch '/home/tmp/somefile': Permission denied
[parallels@fedora ~]$ sudo touch /home/tmp/somefile
[parallels@fedora ~]$ ls -l /home/tmp/somefile
-rw-r--r--. 1 root root 0 Nov  8 20:27 /home/tmp/somefile
[parallels@fedora ~]$ chmod 770 /home/tmp/somefile
chmod: changing permissions of '/home/tmp/somefile': Operation not permitted
[parallels@fedora ~]$ sudo chmod 770 /home/tmp/somefile
[parallels@fedora ~]$ ls -l /home/tmp/somefile
-rwxrwx---. 1 root root 0 Nov  8 20:27 /home/tmp/somefile
[parallels@fedora ~]$
```

The chmod command can be used to set permissions on a file.

Example – 15

```
[parallels@fedora ~]$ sudo chmod u-x /home/tmp/somefile
[parallels@fedora ~]$ ls -l /home/tmp/somefile
-rw-rwx--x. 1 root root 0 Nov  8 20:27 /home/tmp/somefile
[parallels@fedora ~]$
```

You can make relative changes: u-x would remove the owner (u) the ability to execute (x) the file.

Example – 16

```
[parallels@fedora ~]$ sudo chmod o+x /home/tmp/somefile
[parallels@fedora ~]$ ls -l /home/tmp/somefile
-rwxrwx--x. 1 root root 0 Nov  8 20:27 /home/tmp/somefile
[parallels@fedora ~]$
```

You can make relative changes: o+x would add the other user (o) the ability to execute (x) the file.

Example – 17

```
[parallels@fedora ~]$ chmod 660 ~/mysecret
[parallels@fedora ~]$ ls -l ~/mysecret
-rw-rw----. 1 parallels parallels 45 Nov  8 20:22 /home/parallels/mysecret
[parallels@fedora ~]$
```

chmod can set permissions based on absolute octal values, or relative changes. You could use chmod to set permissions on a file based on octet: 660 would give the owner and group rw (read-write), and others no permissions.

Example – 18

```
[parallels@fedora ~]$ su student
Password:
[student@fedora parallels]$ cat /home/parallels/mysecret
cat: /home/parallels/mysecret: Permission denied
[student@fedora parallels]$
```

su command is used to make changes among user.

Example – 19

```
[parallels@fedora ~]$ cat > ~/myshare
It is to be shared with
all the people
[parallels@fedora ~]$ touch ~/myshare
[parallels@fedora ~]$ chmod 666 ~/myshare
[parallels@fedora ~]$ ls -l ~/myshare
-rw-rw-rw-. 1 parallels parallels 39 Nov  8 20:54 /home/parallels/myshare
[parallels@fedora ~]$ kwrite ~/myshare
[parallels@fedora ~]$ cat ~/myshare
It is to be shared with
all the people
[parallels@fedora ~]$ sudo chmod 701 /home/parallels
[parallels@fedora ~]$ su student
Password:
[student@fedora parallels]$ cat /home/parallels/myshare
It is to be shared with
all the people
[student@fedora parallels]$
```

We gave read and write access to the file we created with chmod 666 command. Then we let other users run root directory with chmod 701 command.

Example – 20

```
[parallels@fedora ~]$ cat > ~/mygroupshare
Is it my group share
People in my group can access it
[parallels@fedora ~]$ cat ~/mygroupshare
Is it my group share
People in my group can access it
[parallels@fedora ~]$
```

The cat command is used to read the contents of the file. But if the > sign is placed after the cat command, a new file is created.

Example 21

to add user seyit123 to group liveuser

```
[liveuser@localhost-live Desktop]$ ls -l ~/mygroupshare
-rw-rw-r--. 1 liveuser liveuser 25 Nov  9 13:50 /home/liveuser/mygroupshare
```

```
[liveuser@localhost-live Desktop]$ sudo usermod -a -G liveuser seyit123
```

Now seyit123 added to liveuser group

```
[liveuser@localhost-live Desktop]$ cat /etc/group
```

```
tcpdump:x:72:  
liveuser:x:1000:seyit456,seyit123  
newuser555:x:1001:
```

Example 22

Let's create a group . name is mygroup

```
[liveuser@localhost-live Desktop]$ sudo groupadd mygroup
```

```
[liveuser@localhost-live Desktop]$ cat /etc/group
```

Its here.

```
liveuser:x:1000:se  
newuser555:x:1001:  
seyit456:x:1002:  
seyit123:x:1003:  
mygroup:x:1004:
```

Example 23

To change owner and group of a file

```
[liveuser@localhost-live Desktop]$ sudo chown :mygroup ~/mygroupshare  
[liveuser@localhost-live Desktop]$ ls -l ~/mygroupshare  
-rw-rw-r--. 1 liveuser mygroup 25 Nov  9 13:50 /home/liveuser/mygroupshare  
[liveuser@localhost-live Desktop]$
```

Example 24

Change back mygroupshare group to alex

```
[liveuser@localhost-live Desktop]$ sudo chown :liveuser ~/mygroupshare  
[liveuser@localhost-live Desktop]$ ls -l ~/mygroupshare  
-rw-rw-r--. 1 liveuser liveuser 25 Nov  9 13:50 /home/liveuser/mygroupshare  
[liveuser@localhost-live Desktop]$
```

and give alex group **only read** Access to mygroupshare

```
[liveuser@localhost-live Desktop]$ sudo chmod g-w ~/mygroupshare  
[liveuser@localhost-live Desktop]$ ls -l ~/mygroupshare  
-rw-r--r--. 1 liveuser liveuser 25 Nov  9 13:50 /home/liveuser/mygroupshare  
[liveuser@localhost-live Desktop]$
```

Example 24 – B: Create a new group called “staff”, and create a file that you and a fellow classmate (other user) can collaborate on (both edit). Test whether you have correctly set permissions. Both users should be able to edit the file, yet other users should not have write access.

```
[liveuser@localhost-live Desktop]$ sudo groupadd staff
[liveuser@localhost-live Desktop]$ cat > ~/business
This business file for staff and other user
[liveuser@localhost-live Desktop]$ ls -l ~/business
-rw-rw-r--. 1 liveuser liveuser 44 Nov  9 14:34 /home/liveuser/business

[liveuser@localhost-live Desktop]$ sudo usermod -a -G staff liveuser
[liveuser@localhost-live Desktop]$ cat /etc/group
staff:x:1005:liveuser

[liveuser@localhost-live Desktop]$ ls -l ~/business
-rw-rw-r--. 1 liveuser staff 44 Nov  9 14:34 /home/liveuser/business
```

I didnt understand permission settings in the discription so i did both way i understood.

```
[liveuser@localhost-live Desktop]$ sudo chmod o+x ~/business
[liveuser@localhost-live Desktop]$ ls -l ~/business
-rw-rw-r-x. 1 liveuser staff 44 Nov  9 14:34 /home/liveuser/business
```

Example 25:

Dir ; shows us a list of directory's file and subdirectories

mkdir: creates directory/folder

touch: according to selected file we can make file

```
[liveuser@localhost-live ~]$ dir
business  Documents  Music      Pictures  Templates  tmp
Desktop  Downloads  mygroupshare  Public    test       Videos
[liveuser@localhost-live ~]$ mkdir testtmp
[liveuser@localhost-live ~]$ dir
business  Documents  Music      Pictures  Templates  testtmp  Videos
Desktop  Downloads  mygroupshare  Public    test       tmp
[liveuser@localhost-live ~]$ touch testtmp/test1 testtmp/test2 testtmp/test3
[liveuser@localhost-live ~]$ dir testtmp
test1  test2  test3
[liveuser@localhost-live ~]$
```

Challenge 1

mkdir test: created test folder

touch test/test1 test/test2 test/test3 : creates files in to test folder

From man chmod (chmod information) we found chmod -R (-R means recursively) and changed all permissions of folder's files

```

[liveuser@localhost-live ~]$ mkdir test
[liveuser@localhost-live ~]$ dir
business  Documents  Music      Pictures   Templates  testtmp
Desktop   Downloads  mygroupshare Public     test       Videos
[liveuser@localhost-live ~]$ touch test/test1 test/test2 test/test3
[liveuser@localhost-live ~]$ ls -l test
total 0
-rw-rw-r--. 1 liveuser liveuser 0 Nov  9 15:52 test1
-rw-rw-r--. 1 liveuser liveuser 0 Nov  9 15:52 test2
-rw-rw-r--. 1 liveuser liveuser 0 Nov  9 15:52 test3
[liveuser@localhost-live ~]$ chmod -R 777 test
[liveuser@localhost-live ~]$ ls -l test
total 0
-rwxrwxrwx. 1 liveuser liveuser 0 Nov  9 15:52 test1
-rwxrwxrwx. 1 liveuser liveuser 0 Nov  9 15:52 test2
-rwxrwxrwx. 1 liveuser liveuser 0 Nov  9 15:52 test3
[liveuser@localhost-live ~]$

```

Example 26

umask let us to change default settings of files which is gonna be new created.

My default umask settings

```

[liveuser@localhost-live ~]$ umask -p
umask 0002
[liveuser@localhost-live ~]$ umask -S
u=rwx,g=rwx,o=rx

```

Challenge 2: Using the **umask** builtin command, set your umask so that new files are only rw accessible by you (but not to your group or others):

umask XXX

where XXX is the new umask to use.

Test your new umask value by creating a new file and checking its permissions:

touch newfilename

ls -l newfilename

Do the permissions read “rw-----”? If not, change the umask and try again.

I found umask that is 177.

How i found? I want only rw permissions for user.

So its for user = $r(4)+w(2)=6$

For group = 0

for others= 0

In conclusion its 600 , I extracted it from 777. $777-600=177$ umask.

```
[liveuser@localhost-live ~]$ umask 177
[liveuser@localhost-live ~]$ umask -S
u=rw,g=,o=
[liveuser@localhost-live ~]$ mkdir newfile
[liveuser@localhost-live ~]$ touch newfilename
[liveuser@localhost-live ~]$ ls -l newfilename
-rw-----. 1 liveuser liveuser 0 Nov  9 16:08 newfilename
[liveuser@localhost-live ~]$
```

8.8 Set UID (SUID)

UID: it's original user

EUID: Privileged user has high permissions

Example 27

How to look at Effective UID is specified:

ls -l /usr/bin/passwd

```
[liveuser@localhost-live ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 32712 Jan 30 2021 /usr/bin/passwd
[liveuser@localhost-live ~]$
```

It has "s" so that means its the file UID will be used as EUID

Also, it means UID and EUID are same.

Example 28

statistic about my file

```
[liveuser@localhost-live ~]$ stat /usr/bin/passwd
  File: /usr/bin/passwd
  Size: 32712          Blocks: 64          IO Block: 4096   regular file
Device: fd00h/64768d  Inode: 982           Links: 1
Access: (4755/-rwsr-xr-x)  Uid: (  0/   root)   Gid: (  0/   root)
Context: system_u:object_r:passwd_exec_t:s0
Access: 2021-11-09 00:54:27.617000022 -0500
Modify: 2021-01-30 12:53:22.000000000 -0500
Change: 2021-04-23 07:02:03.559760954 -0400
 Birth: 2021-04-23 07:02:03.554760514 -0400
[liveuser@localhost-live ~]$
```

Gives user id

id -u

```
[liveuser@localhost-live ~]$ id -u
1000
[liveuser@localhost-live ~]$
```

id -u -r

gives EUID

```
[liveuser@localhost-live ~]$ id -u -r
1000
[liveuser@localhost-live ~]$
```

Example 29

If i want to see all the files running with user id equal to root;
ps -af

ps (process stated)= gives info about running processes
af(fully formatted list)=formats precesses as a full list

```
[liveuser@localhost-live ~]$ ps -af
UID          PID    PPID  C STIME TTY          TIME CMD
liveuser     1335     1330  0 12:29 tty2        00:00:00 /usr/libexec/gnome-session-
liveuser     14937    2083  1 17:00 pts/0        00:00:00 ps -af
[liveuser@localhost-live ~]$
```

Example 30

or we can check with ; sudo find -perm -4000 -type f -print

```
alex@Lenovo:~$ sudo find /home -perm -4000 -type f -print
[sudo] password for alex:
/home/alex/accessmysecrets
alex@Lenovo:~$ stat accessmysecrets
  File: accessmysecrets
  Size: 17032          Blocks: 40          IO Block: 4096   regular file
Device: 2h/2d  Inode: 6192449487819206  Links: 1
Access: (4711/-rws--x--x)  Uid: ( 1000/   alex)   Gid: ( 1000/   alex)
Access: 2018-09-29 19:28:36.212452600 +0300
Modify: 2018-09-29 19:28:36.328067700 +0300
Change: 2018-09-29 19:54:57.398230400 +0300
 Birth: -
alex@Lenovo:~$
```

EXAMPLE FROM LAB_SHEET

sudo: super admin

find: help find the files

-perm: means permission

-4000: give us files run with root permission

- type f -print: this funtion same as -af which modify it as list.

9.9. Writing a SUID program in C

Example 31


```
[j_art@fedora ~]$ chmod 600 /home/j_art/mysecret
[j_art@fedora ~]$ ls -l /home/j_art/mysecret
-rw-----. 1 j_art j_art 45 Nov  9 10:21 /home/j_art/mysecret
[j_art@fedora ~]$
```

Making file 'mysecret' accessible only by owner.

Example 32

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <errno.h>
6
7 int main()
8 {
9     printf(" Real UID %d , Real GID %d , Effective UID %d, Effectove GID %d ", getuid(), getgid(),
10    geteuid(), getegid());
11
12     FILE *fp=fopen("mysecret","r");
13     if(fp==NULL)
14     {
15         printf("Error: Could not open file!");
16         exit(EXIT_FAILURE);
17     }
18
19     char c;
20     while((c=getc(fp))!=EOF)
21     {
22         putchar(c);
23     }
24
25     putchar('\n');
26     return EXIT_SUCCESS;
27 }
28
29
```

Creating a accessmysecret.c file and compiling it

```
[j_art@fedora ~]$ cc accessmysecret.c -o accessmysecret
[j_art@fedora ~]$
```

Example 33

```
[j_art@fedora ~]$ chmod u+s accessmysecret
[j_art@fedora ~]$ ls -l accessmysecret
-rws--x--x. 1 j_art j_art 25704 Nov  9 15:01 accessmysecret
[j_art@fedora ~]$ stat mysecret
  File: mysecret
  Size: 45          Blocks: 8          IO Block: 4096   regular file
Device: 26h/38d Inode: 940          Links: 1
Access: (0600/-rw-----)  Uid: ( 1000/   j_art)   Gid: ( 1000/   j_art)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2021-11-09 10:30:22.951850642 +0300
Modify: 2021-11-09 10:21:52.586563727 +0300
Change: 2021-11-09 14:39:23.899378106 +0300
 Birth: 2021-11-09 10:21:29.810595817 +0300
[j_art@fedora ~]$
```

Setting file permission to setuid with command chmod and checking permission of the file.

Example 34

```
[j_art@fedora ~]$ chmod u+s accessmysecret
[j_art@fedora ~]$ ls -l accessmysecret
-rws--x--x. 1 j_art j_art 25704 Nov  9 15:01 accessmysecret
[j_art@fedora ~]$ ./accessmysecret
Real UID 1000 , Real GID 1000 , Effective UID 1000, Effectove GID 1000 It is my secret
Here it is
My secret is here
[j_art@fedora ~]$
```

Running the program

Example 35

```
[j_art@fedora ~]$ su user2
Password:
[user2@fedora j_art]$ ./accessmysecret
Real UID 1001 , Real GID 1001 , Effective UID 1000, Effectove GID 1001 It is my secret
Here it is
My secret is here
[user2@fedora j_art]$
```

Running the program under different user. The uid and gid id is different than the one before

Challenge 3

Challenge 4

```
if (getuid() != geteuid()){
    printf("You're not allowed here");
    exit(EXIT_FAILURE);
}
```

By adding if condition to the beginning of our program we can put restrictions to our file.

```
Real UID 1000 , Real GID 1000 , Effective UID 1000, Effectove GID 1000 It is my secret
Here it is
My secret is here

[j_art@fedora ~]$ su student
Password:
[student@fedora j_art]$ ./test
You're not allowed here[student@fedora j_art]$
```

Challenge 5

```

if (getuid() == geteuid())
{
    while((c=getc(fp)) != EOF)
    {
        putchar(c);
    }

    else
    while((c=getc(fp)) != '\n')
    {
        putchar(c);
    }
}

```

Same as challenge 4, we can use if-else condition to print different contents of our file.

```

[j_art@fedora ~]$ ./test2
Real UID 1000 , Real GID 1000 , Effective UID 1000, Effectove GID 1000 It is my secret
Here it is
My secret is here

[j_art@fedora ~]$ su student
Password:
[student@fedora j_art]$ ./test2
Real UID 1002 , Real GID 1002 , Effective UID 1000, Effectove GID 1002 It is my secret
[student@fedora j_art]$ █

```

Challenge 6

```

if (getuid() != 1002)
{
    printf("\n ONLY specific user is allowed in here\n");
    exit(EXIT_FAILURE);
}

```

For this challenge we use if condition again. In this example we gave user “student” with UID=1002 to access to our file.

```

[j_art@fedora ~]$ ./test

ONLY specific user is allowed in here
[j_art@fedora ~]$ su student
Password:
[student@fedora j_art]$ ./test
Real UID 1002 , Real GID 1002 , Effective UID 1000, Effectove GID 1002 It is my secret
Here it is
My secret is here

[student@fedora j_art]$ █

```

10 Linus extended ACLs

Example 36

```
[j_art@fedora ~]$ setfacl -m u:student:r ~/mysecret
[j_art@fedora ~]$ getfacl ~/mysecret
getfacl: Removing leading '/' from absolute path names
# file: home/j_art/mysecret
# owner: j_art
# group: j_art
user::rw-
user:student:r--
group::---
mask::r--
other::---
```

We used setfacl command to give student user read permission of “mysecret” file. And with getfacl on our file we can check who has which kind of permission on our file.