Practical 11: ACL Permission and Backup

Practice 11.1: Sharing a File with Additional User

To explore how ACLs can be used to grant an additional user access to a file, try the following (**Create a guest account using useradd and set its password as redhat, if the guest account does not exist**,):

1. Login to the Redhat Client as **student** (or any ordinary account) and open a terminal window.
2. Create a user called **guest** with the password "**redhat**" in the Client machine. How do you verify the user creation is successful? (see Annex A)
3. Create (and verify) a file named /tmp/acl-test by issuing the following command:

```
[student@client ~]# echo student > /tmp/acl-test
```

```
[student@client ~]# ls -l /tmp/acl-test
-rw-rw-r--      1 student       student         1 Oct 21 09:43  /tmp/acl-test
```

4. Login as **guest**, try to read and then append some data to that same file (it should fail to write), then return to student:

```
[student@client ~]$ su – guest
Password: redhat
[guest@client ~]$ cat /tmp/acl-test
Student
[guest@client ~]$ echo guest > /tmp/acl-test
-bash: /tmp/acl-test: permission denied
[guest@client ~]$ exit
```

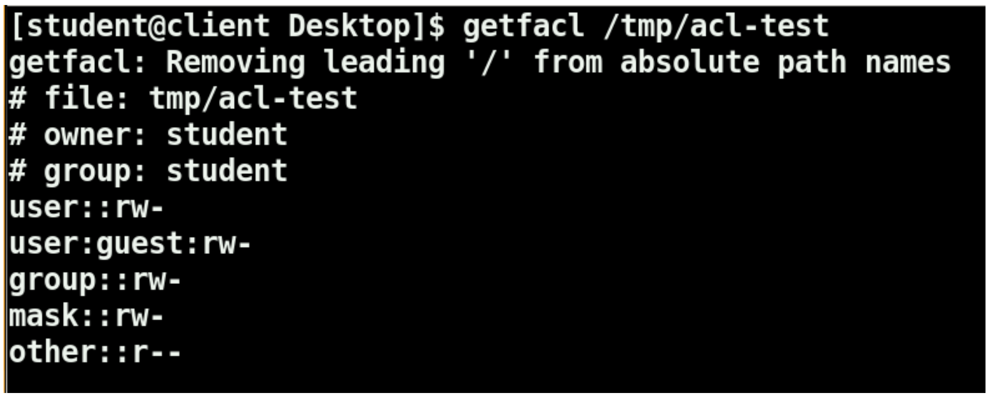
5. As student, we will grant read and write permission to user **guest** to our file by using the command:

```
[student@client ~]$ setfacl –m u:guest:rw /tmp/acl-test
```

6. As guest, try to append some data to that same file again (it should now succeed) and view the changes:

```
[student@client ~]$ su –guest
Password: password
[guest@client ~]$ echo guest >> /tmp/acl-test
[guest@client ~]$ cat /tmp/acl-test
Student
Guest
[guest@client ~]$
```

7. Look at the /tmp/acl-test ACL settings by issuing the getfacl command.

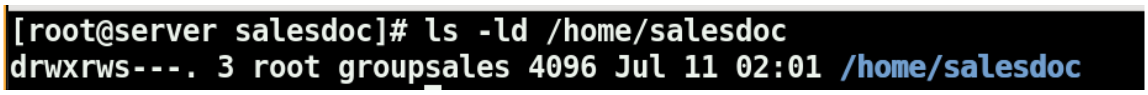


```
[student@client Desktop]$ getfacl /tmp/acl-test
getfacl: Removing leading '/' from absolute path names
# file: tmp/acl-test
# owner: student
# group: student
user::rw-
user:guest:rw-
group::rw-
mask::rw-
other::r--
```

8. When done, close your terminal window.

Practice 11.2: Share a Collaborative Directory using ACL

In the previous practical (10A), on the Redhat Server we created users sales1, sales2, sales3 and mktg We also created a group groupsales and added sales1, sales2 and sales3 as its members. A shared directory /home/salesdoc was configured as shown below:



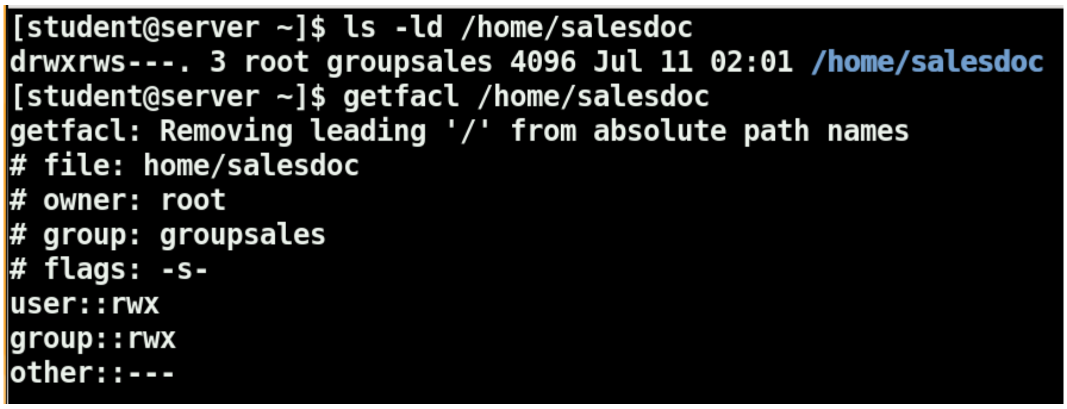
```
[root@server salesdoc]# ls -ld /home/salesdoc
drwxrws---. 3 root groupsales 4096 Jul 11 02:01 /home/salesdoc
```

Make sure you have the above before proceeding.

.

1. Login to the Redhat Client as **student**. Use command **ssh** to log into the Redhat Server as student and escalate your privileges to root with **su -**.

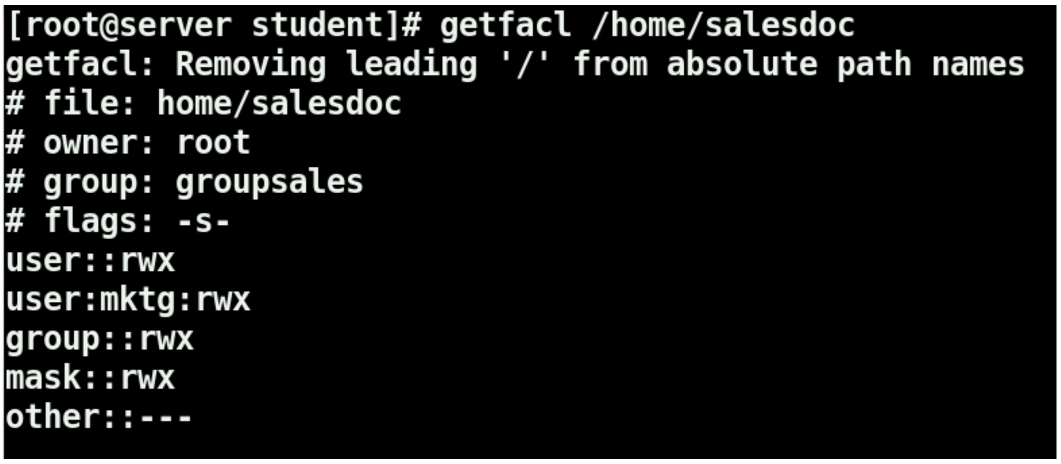
2. Run the 2 commands **ls -ld /home/salesdoc** and **getfacl /home/salesdoc,** and compare their outputs.



```
[student@server ~]$ ls -ld /home/salesdoc
drwxrws---. 3 root groupsales 4096 Jul 11 02:01 /home/salesdoc
[student@server ~]$ getfacl /home/salesdoc
getfacl: Removing leading '/' from absolute path names
# file: home/salesdoc
# owner: root
# group: groupsales
# flags: -s-
user::rwx
group::rwx
other::---
```

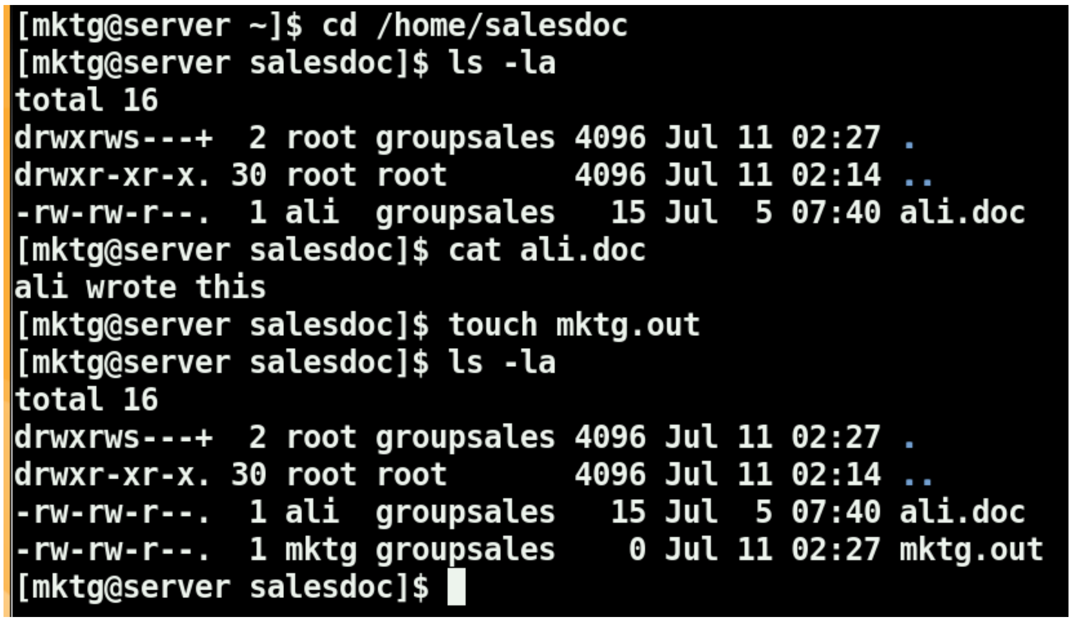
3. Use setfacl to give mktg user the Read, Write and Execute permission to the /home/salesdoc (refer to 11.1 s/no. 5 if you cannot recall)

   Upon successful execution, you should see below when you display the ACL:



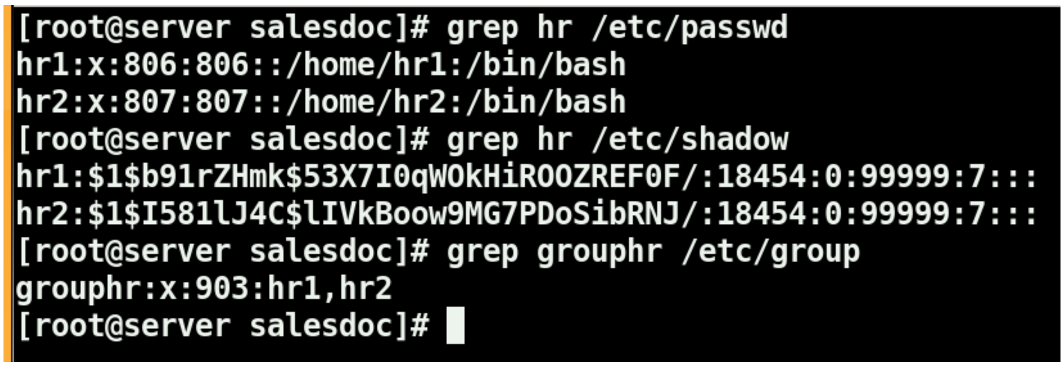
```
[root@server student]# getfacl /home/salesdoc
getfacl: Removing leading '/' from absolute path names
# file: home/salesdoc
# owner: root
# group: groupsales
# flags: -s-
user::rwx
user:mktg:rwx
group::rwx
mask::rwx
other::---
```

4. Switch to user mktg and try to change directory to /home/salesdoc, read the files there, and create some files.



```
[mktg@server ~]$ cd /home/salesdoc
[mktg@server salesdoc]$ ls -la
total 16
drwxrws---+  2 root groupsales 4096 Jul 11 02:27 .
drwxr-xr-x. 30 root root       4096 Jul 11 02:14 ..
-rw-rw-r--.  1 ali  groupsales   15 Jul  5 07:40 ali.doc
[mktg@server salesdoc]$ cat ali.doc
ali wrote this
[mktg@server salesdoc]$ touch mktg.out
[mktg@server salesdoc]$ ls -la
total 16
drwxrws---+  2 root groupsales 4096 Jul 11 02:27 .
drwxr-xr-x. 30 root root       4096 Jul 11 02:14 ..
-rw-rw-r--.  1 ali  groupsales   15 Jul  5 07:40 ali.doc
-rw-rw-r--.  1 mktg groupsales    0 Jul 11 02:27 mktg.out
[mktg@server salesdoc]$
```

5. Create another group called grouphr and add 2 users to it – hr1 and hr2. You need to create the 2 users too. Use ACL to give this group Read only access to the /home/salesdoc directory.

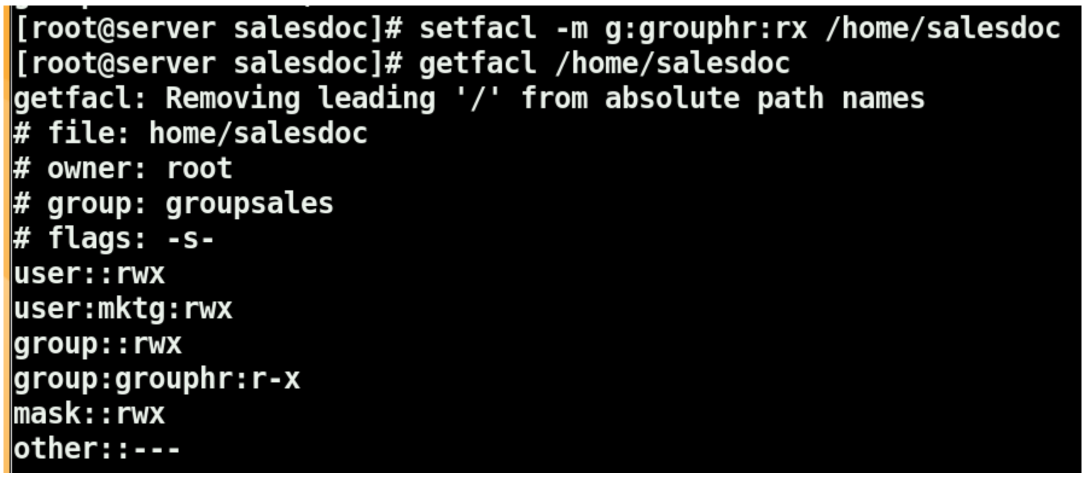


```
[root@server salesdoc]# grep hr /etc/passwd
hr1:x:806:806::/home/hr1:/bin/bash
hr2:x:807:807::/home/hr2:/bin/bash
[root@server salesdoc]# grep hr /etc/shadow
hr1:$1$b91rZHmk$53X7I0qWOkHiROOZREF0F/:18454:0:99999:7:::
hr2:$1$I581lJ4C$lIVkBoow9MG7PDoSibRNJ/:18454:0:99999:7:::
[root@server salesdoc]# grep grouphr /etc/group
grouphr:x:903:hr1,hr2
[root@server salesdoc]#
```

The figure above shows the successful creation of 2 users hr1 and hr2 (and their passwords), a grouphr with hr1 and hr2 as its members.

6. Below shows the adding of grouphr to have read permission to /home/salesdoc via ACL.

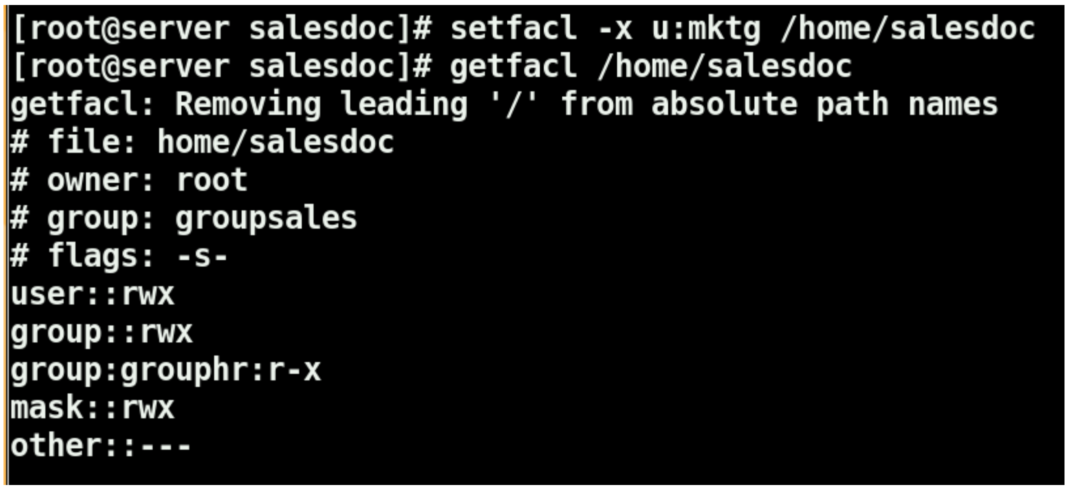
```
setfacl -m g:grouphr:rx /home/salesdoc
```



```
[root@server salesdoc]# setfacl -m g:grouphr:rx /home/salesdoc
[root@server salesdoc]# getfacl /home/salesdoc
getfacl: Removing leading '/' from absolute path names
# file: home/salesdoc
# owner: root
# group: groupsales
# flags: -s-
user::rwx
user:mktg:rwx
group::rwx
group:grouphr:r-x
mask::rwx
other::---
```

7. Switch to hr1 account and attempt to access the /home/salesdoc directory. What actions can hr1 perform there?

8. We want to remove mktg user permission from /home/salesdoc directory. Try the commands below.



```
[root@server salesdoc]# setfacl -x u:mktg /home/salesdoc
[root@server salesdoc]# getfacl /home/salesdoc
getfacl: Removing leading '/' from absolute path names
# file: home/salesdoc
# owner: root
# group: groupsales
# flags: -s-
user::rwx
group::rwx
group:grouphr:r-x
mask::rwx
other::---
```

9. When done, close your **ssh** session and terminal window.

Practice 11.3: Backup and Restore using tar

1) Login to the Client as student and open a terminal window.

2) Remote login to the Server using ssh (if you are unable to ssh, then log in to the server directly).

3) At the server, switch to root (su -)

4) Backup the /home directory to /tmp/home.tar

```
# tar -cvpzf /tmp/home.tar /home
```

5) List the archive file with **`ls -l /tmp/home.tar`**

6) Create a directory to contain the extracted files.

```
# mkdir /tmp/extracted
```

7) Extract the archive into the directory.

```
# tar -xvzf /tmp/home.tar -C /tmp/extracted
```

8) Check /tmp/extracted to see if the extraction was successful.

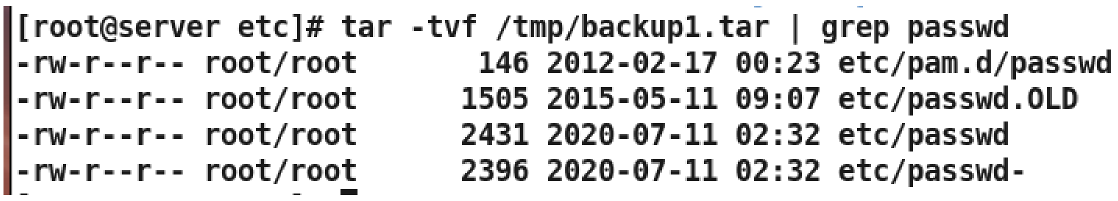
9) You can archive several directories at the same time. For example, we shall archive the /etc and /bin directories

```
# tar -cvpzf /tmp/backup1.tar /etc /bin
```

10) We can extract a single file from the archive. For example, we wish to restore the passwd file from the archive.

(a) First we see if the archive has the file:

```
# tar -tvf /tmp/backup1.tar | grep passwd
```

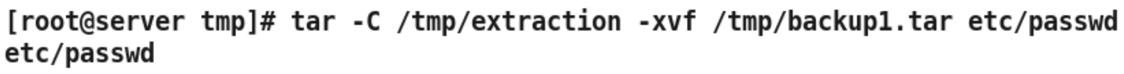


```
[root@server etc]# tar -tvf /tmp/backup1.tar | grep passwd
-rw-r--r-- root/root       146 2012-02-17 00:23 etc/pam.d/passwd
-rw-r--r-- root/root      1505 2015-05-11 09:07 etc/passwd.OLD
-rw-r--r-- root/root      2431 2020-07-11 02:32 etc/passwd
-rw-r--r-- root/root      2396 2020-07-11 02:32 etc/passwd-
```

(b) After finding the file, we can extract it out to another location (/tmp/extraction).

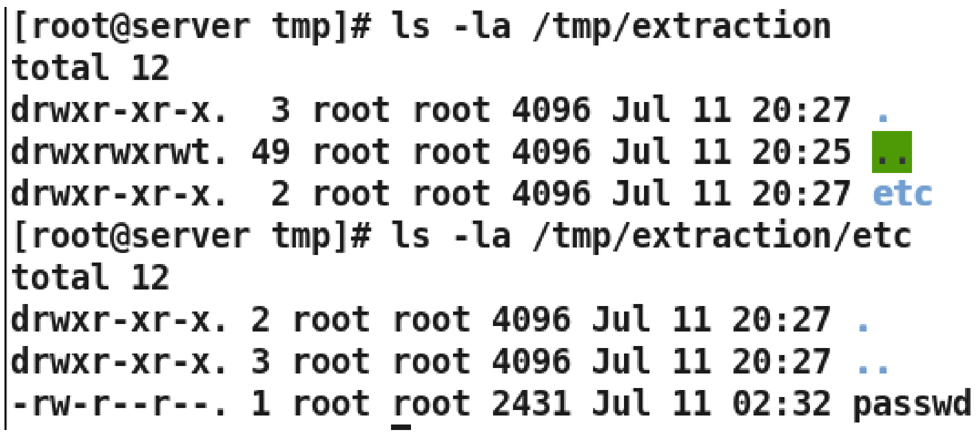
```
# mkdir /tmp/extraction
# tar -C /tmp/extraction -xvf /tmp/backup1.tar etc/passwd
```

If successful you shall see below display:



```
[root@server tmp]# tar -C /tmp/extraction -xvf /tmp/backup1.tar etc/passwd
etc/passwd
```

(c) List /tmp/extraction to see if the passwd file is there.



```
[root@server tmp]# ls -la /tmp/extraction
total 12
drwxr-xr-x.  3 root root 4096 Jul 11 20:27 .
drwxrwxrwt. 49 root root 4096 Jul 11 20:25 ..
drwxr-xr-x.  2 root root 4096 Jul 11 20:27 etc
[root@server tmp]# ls -la /tmp/extraction/etc
total 12
drwxr-xr-x. 2 root root 4096 Jul 11 20:27 .
drwxr-xr-x. 3 root root 4096 Jul 11 20:27 ..
-rw-r--r--. 1 root root 2431 Jul 11 02:32 passwd
```

Practice 11.4: Backup and Restore using rsync

1) Login to the Client as student and open a terminal window.
2) Remote login to the Server using ssh (if you are unable to ssh, then log in to the server directly).
3) At the server, switch to root (su -)
4) Backup the /home directory to /tmp/backuphome using rsync

   ```
   # rsync -av /home/ /tmp/backuphome
   ```
5) Change to /tmp/backuphome to view the files.
6) Create a new directory in /home

   ```
   # mkdir /home/newdir
   ```
7) Create a new file in /home/student

   ```
   # touch /home/student/newfile.out
   ```
8) Do the backup in step 4 again. Do you see the difference of the output of step 4 and step 8?

[The End]

Reference Notes

<u>Access Control Lists (ACLs)</u>

ACLs can be used in some special circumstances, like when you need to define different permission levels for two (or more) users because they do not and cannot be part of the same group, or you cannot change the group ownership. ACLs grant or deny actions very much like standard permissions do, but allow you to define permissions for multiple users and groups.

<u>Viewing and Managing ACLs</u>

For Viewing :
# **getfacl filename**

For Modifying (Adding or Changing):
#**setfacl –m u:gandalf: rw filename**

For Removing (Expunging):
#setfacl –x u:gandalf filename

To view the access control list (ACL) for a file, use the **getfacl** command:

```
[student@client ~]# getfacl /shared/schedule.txt
getfacl: removing leading '/' from absolute path names

# file: shared/schedule.txt
# owner: student
# group: student

user::rw-
user:bob:rwx
group::r—
group:admins:rw-
mask::rwx
other::r--
```

Note that you can quickly see if there are ACLs defined on a file (or directory) looking at the **ls –l** output, you will get a "+" sign appearing at the end of the permission columns.

```
[student@client ~]# ls –l /shared
-rw-rwx-r--+   1   student   student     0 2009-06-25 00:52 schedular.txt
```

Please also note that permissions defined for the group owner are not shown anymore by the **ls –l** output! What you get in the second triplet is the same value that getfacl is showing on the "mask::" line of its output. That value is the maximum permission definable for the group owner or for any of the users or groups managed by an ACL.

ACLs can be modified using the **setfacl** command.

Group Collaboration:

Basic understanding :

- ACLs for group use "g:" instead of "u:"
- Automatic ACL setting
- New files inherit default ACL (if set) from directory

Command Format:

# setfacl –m d:g:groupname:rw directory

Default for groups can share files with multiple groups

Default ACLs define what be the ACLs for newly created files and subdirectories.

New files (either copied into a directory or newly created within) where a default has been set will inherit the default ACL from the directory.

Be careful, as settings a default does not alter permissions on the *existing files nor on the directory itself.*

```
[root@client ~]# ls -ld /shared/
drwxrwsr-x 2 root nypdir 4096 2009-06-25 11:07 /shared
[root@client ~]# setfacl –m d:g:goodpeople:rw /shared/
[root@client ~]# getfacl /shared/
Getfacl: Removing leading '/' from absolute path names
# file: shared
# owner:  root
# group: nypdir

User::rwx
Group:: rwx
Other ::r_x
Default:user::rwx
Default:group::rwx
```

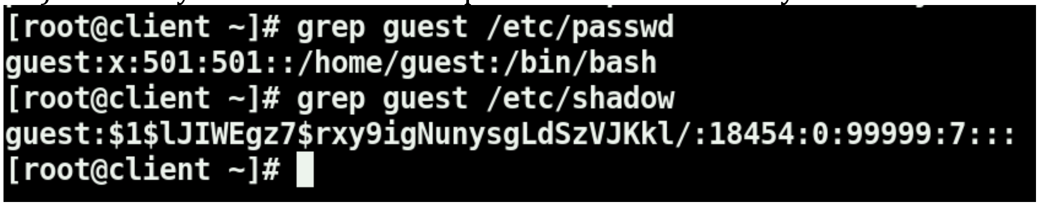
```
Default:group:goodpeople:rw-
Default:mask::rwx
Default:other::r-x
[root#client ~]# touch /shared/third
[root#client ~]# getfacl /shared/third

Get facl: removing leading '/' from absolute path names
# file: shared/third
# owner:  root
# group: nypdir
User::rw-
Group:: rwx    #effective:rw-
Group:goodpeople:rw-
Mask::rw-
Other ::r_-
```

Notice that the new file has nypdirset as group owner, because of the SGID bit on the shared directory and also an ACL defined for the group goodpeople which will allow the members to read and write the file. The effective:rw- is telling us that group owner will not have execute permissions on the file, because of the mask:: rw-. Do not forgot the **mv** command and the **cp** command as well when used with the –p option, will preserve permissions which have been set on a file, ignoring any default ACLs.

Annex A

1) To verify a user account and password is successfully created:



```
[root@client ~]# grep guest /etc/passwd
guest:x:501:501::/home/guest:/bin/bash
[root@client ~]# grep guest /etc/shadow
guest:$1$lJIWEgz7$rxy9igNunysgLdSzVJKkl/:18454:0:99999:7:::
[root@client ~]#
```