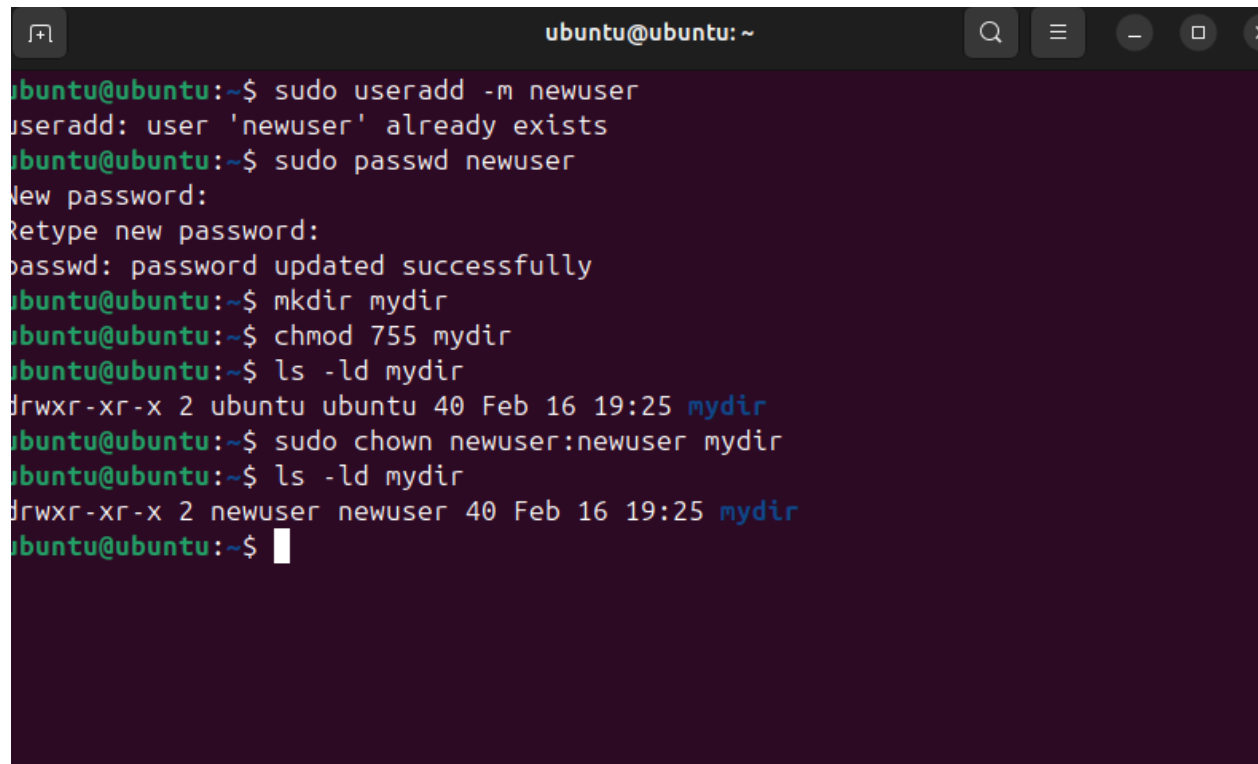


Lucy Njuguna

Access Control in Linux

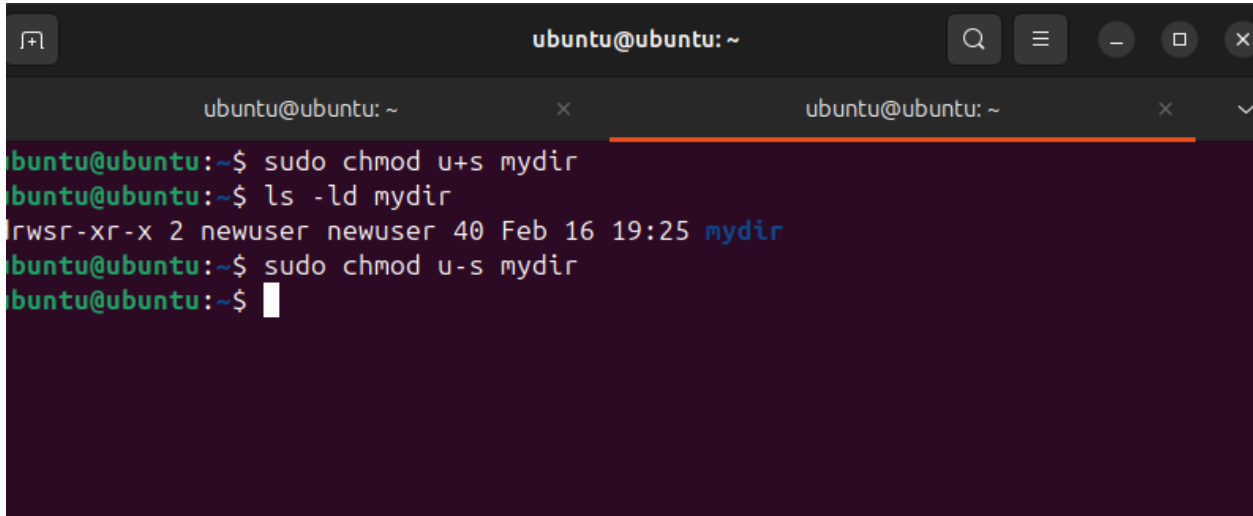
1. User and Group Permissions:

A terminal window titled 'ubuntu@ubuntu: ~' with standard Ubuntu window controls. The terminal shows a series of commands to create a user, set a password, create a directory, set permissions, and change ownership. The output shows that the user 'newuser' already exists, the password is successfully updated, the directory 'mydir' is created with permissions 'drwxr-xr-x', and its ownership is changed to 'newuser:newuser'.

```
ubuntu@ubuntu:~$ sudo useradd -m newuser
useradd: user 'newuser' already exists
ubuntu@ubuntu:~$ sudo passwd newuser
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ubuntu:~$ mkdir mydir
ubuntu@ubuntu:~$ chmod 755 mydir
ubuntu@ubuntu:~$ ls -ld mydir
drwxr-xr-x 2 ubuntu ubuntu 40 Feb 16 19:25 mydir
ubuntu@ubuntu:~$ sudo chown newuser:newuser mydir
ubuntu@ubuntu:~$ ls -ld mydir
drwxr-xr-x 2 newuser newuser 40 Feb 16 19:25 mydir
ubuntu@ubuntu:~$
```

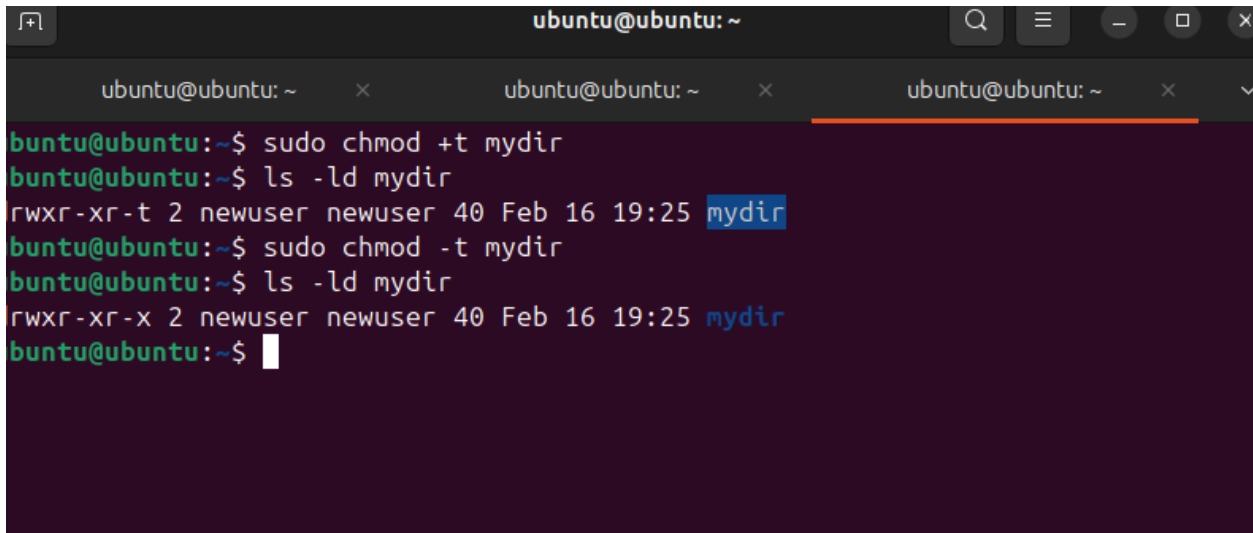
2. File Attributes:

Set and unset the setuid



```
ubuntu@ubuntu: ~  
buntu@ubuntu:~$ sudo chmod u+s mydir  
buntu@ubuntu:~$ ls -ld mydir  
rwsr-xr-x 2 newuser newuser 40 Feb 16 19:25 mydir  
buntu@ubuntu:~$ sudo chmod u-s mydir  
buntu@ubuntu:~$
```

setgid, and sticky bit



```
ubuntu@ubuntu: ~  
buntu@ubuntu:~$ sudo chmod +t mydir  
buntu@ubuntu:~$ ls -ld mydir  
rwxr-xr-t 2 newuser newuser 40 Feb 16 19:25 mydir  
buntu@ubuntu:~$ sudo chmod -t mydir  
buntu@ubuntu:~$ ls -ld mydir  
rwxr-xr-x 2 newuser newuser 40 Feb 16 19:25 mydir  
buntu@ubuntu:~$
```

3. Access Control Lists (ACLs):

- Create a new file and assign a basic ACL to it.

A terminal window titled 'ubuntu@ubuntu: ~' with standard window controls. The terminal shows the following commands and output:

```
ubuntu@ubuntu:~$ touch myfile.txt
ubuntu@ubuntu:~$ sudo setfacl -m u:newuser:rw myfile.txt
ubuntu@ubuntu:~$ getfacl myfile.txt
# file: myfile.txt
# owner: ubuntu
# group: ubuntu
user::rw-
user:newuser:rw-
group::rw-
mask::rw-
other::r--

ubuntu@ubuntu:~$
```

Modify the ACL to grant or revoke specific permissions for a user or group.

```
ubuntu@ubuntu: ~  
buntu@ubuntu:~$ sudo setfacl -m u:newuser:rwx myfile.txt  
buntu@ubuntu:~$ sudo setfacl -m g:developers:r myfile.txt  
setfacl: Option -m: Invalid argument near character 3  
buntu@ubuntu:~$ sudo setfacl -m g:users:r myfile.txt  
buntu@ubuntu:~$ getfacl myfile.txt  
# file: myfile.txt  
# owner: ubuntu  
# group: ubuntu  
user::rw-  
group::rw-  
other::r--  
  
buntu@ubuntu:~$ sudo setfacl -x u:newuser myfile.txt  
buntu@ubuntu:~$ sudo setfacl -b myfile.txt  
buntu@ubuntu:~$
```

Investigation Report: Access Control in Linux

Introduction

Access control is an important aspect of Linux security, because it ensures that files and directories are protected from unauthorized access and modifications. Properly managing permissions helps maintain system integrity by allowing only authorized users to read, write, or execute files. This report explores key access control methods, including standard file permissions, special attributes, and Access Control Lists (ACLs).

User and Group Permissions

Linux file permissions are based on three categories: owner, group, and others, each with three permission types: read (r), write (w), and execute (x). The `chmod` command is used to modify these permissions, while `chown` changes file ownership. In the case above, the file owner has full access, while the group and others can only read and execute it. Improper permission settings can either restrict necessary access or expose files to unauthorized users; hence, the correct balance should be met.

File Attributes: `setuid`, `setgid`, and Sticky Bit

Beyond standard permissions, Linux provides additional file attributes to control execution and access:

- **setuid (`chmod u+s`):** Ensures that a file executes with the permissions of its owner rather than the user running it. This is commonly used in system utilities like `passwd`.
- **setgid (`chmod g+s`):** Ensures files created in a directory inherit the directory's group ownership, making it useful for shared project directories.
- **Sticky Bit (`chmod +t`):** Prevents users from deleting files owned by others in shared directories, such as `/tmp`.



Access Control Lists (ACLs)

ACLs provide a more flexible permission model by allowing specific users or groups to have customized access beyond traditional Linux permissions. The `setfacl` command assigns ACLs, while `getfacl` displays current ACL settings.

Example: Granting read and write permissions to `newuser` in the case above the following command was used: **`setfacl -m u:newuser:rw myfile.txt`**

ACLs are mostly useful in environments where multiple users require different levels of access to the same files or directories.

Conclusion

Effective access control is essential for maintaining security and organization in Linux systems. Standard file permissions, special attributes, and ACLs work together to provide a layered approach to access management. Understanding and implementation of these methods ensures that data remains protected while allowing necessary access for users and groups.