Lab 3 - Linux Fundamentals Contd.

Links in Linux:

Similar to shortcuts in Windows, the same are called `links` in Linux. These can be created using the `ln` command and after denoted by `l` as the sticky bit.

Linux Basic Commands

`sudo`

`sudo` stands for `Super User Do`. It allows you to run commands with elevated privileges. Similar to `Run as Administrator` in Windows.

```
sudo <command>
```

`su`

`su` stands for `Switch User`. It allows you to switch to another user account.

```
su <username>
```

`passwd`

`passwd` allows you to change your password.

```
passwd
```

`chmod`

`chmod` stands for `Change Mode`. It allows you to change the permissions of a file or directory.

```
chmod <permissions> <file>
```

▼ **PERMISSIONS:**

Permissions of a file in linux can be viewed when pressed with `ls -l`

`chown`

`chown` stands for `Change Owner`. It allows you to change the owner of a file or directory.

```
chown <owner> <file>
```

```
Usage:
chown <username>:<group-name> <file-name>
```

useradd

`useradd` allows you to create a new user account (non-interactive)

```
useradd <username>
```

adduser

`adduser` allows you to create a new user account

```
adduser <username>
```

userdel

`userdel` allows you to delete a user account.

```
userdel <username>
```

groupadd

`groupadd` allows you to create a new group.

```
groupadd <groupname>
```

groupdel

`groupdel` allows you to delete a group.

```
groupdel <groupname>
```

apt

`apt` stands for `Advanced Packaging Tool`. It is a package manager for Debian-based Linux distributions.

```
# In order to install a software
sudo apt install <package-name>

# In order to update the package list
sudo apt update
```

`wget`

`wget` stands for `Web Get`. It is a command-line utility for downloading files from the internet.

```
wget <url>
```

`curl`

`curl` stands for `Client URL`. It is a command-line utility for transferring data to or from a server.

```
# In order to download a file
curl -O <url>
```

`history`

`history` displays the history of commands that have been executed in the current shell.

```
history
```

Permissions in Linux

In order to list the permission, we'll use `ls` command with `-l` flag.

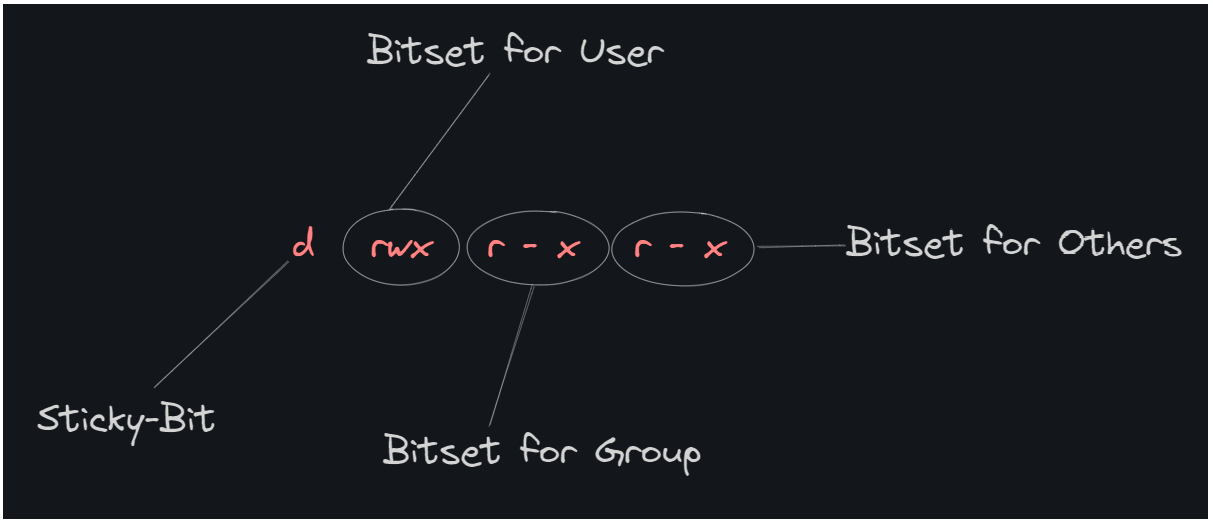
The output inside the `~` is as follows:

```
drwxr-xr-x 1 theflash2k admin  4096 Mar 21  2023 Documents
-rw-r--r-- 1 theflash2k admin    16 Feb 19  2023 Downloads
-rwxr-xr-x 1 theflash2k admin 16328 Feb 19  2023 nulled
-rw-r--r-- 1 theflash2k admin   594 Feb 19  2023 nulled.c
-rw-r--r-- 1 theflash2k admin    24 Feb 19  2023 old_flag.txt
drwxr-xr-x 1 theflash2k admin  4096 Mar 12  2023 pwnable
drwxr-xr-x 1 theflash2k admin  4096 Feb 25  2023 example.sh
```

Each entry has a specific meaning

```
drwxr-xr-x
```

are the permissions. They are divided as follows:



The sticky-bit can vary, `-d` is for directories, `-l` is for links, `-s` is `setuids` etc.

`r` stands for READ

`w` stands for WRITE

`x` stands for EXECUTE

These can also be represented by numbers.

```
r => 4
w => 2
x => 1
```

When trying to set permission for a single file using `chmod`, we can specify the number in all three sets. (***USER***, ***GROUP*** and ***OTHERS***). The singular permissions will add up to give a single number. Meaning, in the bitset for user, if we want the user to have both read and write permission, our number will be (4 + 2) = 6. Hence, 6. Then, if we want the group to have only write and execute, then we will add (2 + 1) = 3. Then, for others, we don't want others to have any permission, so we will simply type 0. Therefore, the final command will become:

```
chmod 630 <file>
```

Similar to numbers, we can easily give and take permissions using `characters` in the chmod command. The character set is as follows:

```
chmod <bitset><give/take><permission>
Like, in the bitset, the possible characters are:

u -> for user
```

```
g -> for group
o -> for others.

These can be used individually or can be chained together as well.

Then, in give/take, we simply type `+` if we want to give a permission or `-` if we want to take it

Then, the permissions allowed are:

r -> read
w -> write
x -> execute.

Similar to bitset, these can be chained as well.
```

Example 1:

File `/bin/test` is owned by `test` and group-owned by `test-grp`. I want to `test` to be able to read, write and execute. `test-grp` to only have read and execute permission and others to have `read` permission.

For this example, we can easily divide it like this:

```
# User:
read (4), write(2), execute(1) => 7
read (4), execute(1)           => 5
read (4)                       => 4
Hence, the final permission set becomes: 754.
```
