Name: Basant Tarik Salah

Instructor: Eng. Mohamed Abo-Khalil

SIC7_Task.Phase3

Part1: Tasks

1. Create a new group iot_team and add your user to it

```
basant@basant-VirtualBox:~$ sudo groupadd IOT_Team
basant@basant-VirtualBox:~$ sudo usermod -aG iot_team $USER
```

2. Create a new developer user, add it to the group.

```
basant@basant-VirtualBox:~$ sudo adduser developer
Adding user `developer' ...
Adding new group `developer' (1001) ...
Adding new user `developer' (1001) with group `developer' ...
The home directory `/home/developer' already exists.  Not copying from `/etc/skel'.
adduser: Warning: The home directory `/home/developer' does not belong to the user you are currently creating.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for developer
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
basant@basant-VirtualBox:~$
basant@basant-VirtualBox:~$ sudo usermod -aG IOT_Team developer
```

3. Change ownership of iot_logger to the developer + group.

```
basant@basant-VirtualBox:~$ sudo chown -R developer:IOT_Team iot_logger
basant@basant-VirtualBox:~$
```

4. Set permissions: group can read/write logs, others blocked.

```
basant@basant-VirtualBox:~$ sudo chmod ug+rwx iot_logger/logs
basant@basant-VirtualBox:~$ sudo chmod o-rwx iot_logger/logs
```

Or

```
basant@basant-VirtualBox:~$ sudo chmod -R 770 iot_logger/logs
```

5. Test access as new user, then remove test user.

```
basant@basant-VirtualBox:~$ su - developer
Password:
developer@basant-VirtualBox:~$ cd /home/basant/iot_logger/logs
developer@basant-VirtualBox:/home/basant/iot_logger/logs$
```

```
developer@basant-VirtualBox:/home/basant/iot_logger/logs$ exit
logout
basant@basant-VirtualBox:~$ sudo deluser developer
Removing user `developer' ...
Warning: group `developer' has no more members.
Done.
basant@basant-VirtualBox:~$
```

Part 2: Open Ended Questions

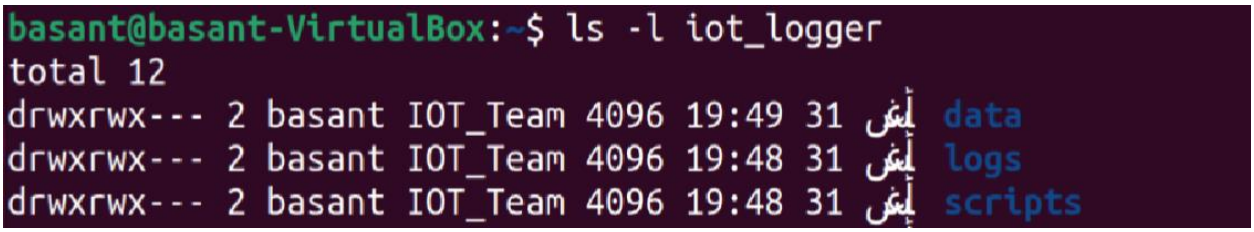
1. **How do Linux file permissions (r, w, x) work for files vs directories? Give an example using ls -l.**

Ans.

Linux uses 3 letters to control what users can do:

- **r = read**
  - For a **file**: enables opening file and seeing its content.
  - For a **directory:** enables listing its inside files.
- **w = write**
  - For a **file:** enables editing or deleting it.
  - For a **directory:** enables creating new files or deleting the existing files.
- **x = execute**
  - For a **file:** enables running it
  - For a **directory:** enables entering it

For Example:



rwx: the owner can do can do everything (read, write, execute).
rwx: the group members can also do everything.
---: others have no permission at all. They can't even look inside

2. Explain octal notation for permissions and what the umask command does. Give one calculation example.

Ans.

Permissions can be written in octal notation as numbers instead of letters.
Each permission has a number:

- **r "read" = 4**
- **w "write" = 2**
- **x "execute" = 1**

They are added together:

- **rwx = 7** (full access)
- **rw- = 6** (read + write)
- **r-x = 5** (read + execute)
- **--- = 0** (no permissions)

For Example **chmod 770** means:

- Owner = 7 (rwx)
- Group = 7 (rwx)
- Others = 0 (no access)

umask sets the default *permissions* when new files are created. It subtracts its value from the default file value.

For Example:

Default file mode = 666 (rw-rw-rw-)

If umask = 022: new file = 644 (rw-r--r--)

- Owner can read/write.
- Group and others can only read.

3. What is the difference between the root user and a normal user? Why root is considered dangerous?

Ans.

- **Normal users**

They have a limited power. As they can only change files in their home folder and can't affect system files directly.

- **Root users (super users)**

They have unlimited power. As they can do anything without restrictions. They can install or remove software, delete any file, manage users and change system settings.

The command: **sudo** gives temporary root power.

It is dangerous because mistakes can break the whole system.