Report

1. **Create Users**

```
[student@localhost ~]$ sudo useradd Supervisor1
[sudo] password for student:
[student@localhost ~]$ sudo useradd Engineer1
[student@localhost ~]$ sudo useradd Engineer2
[student@localhost ~]$ sudo useradd Engineer3
[student@localhost ~]$ sudo useradd Operator1
[student@localhost ~]$ sudo useradd Operator2
[student@localhost ~]$ sudo useradd Operator3
```

Create 1 supervisor and 6 users (3 engineers and 3 operators) using sudo command that allow a non-root user to execute commands with superuser privileges. (if it is in the list of sudoers).

2. **Add Supervisor1 to the list of sudoers (wheel group)**

```
[student@localhost ~]$ sudo usermod -aG wheel Supervisor1
[student@localhost ~]$
```

Add the newly created supervisor to the list of sudoers (wheel group) for later administrative usage.

- **sudo**: Executes the command with superuser privileges.
- **usermod**: Modifies user account properties.
- **-aG**: Adds the user to the specified supplementary group (wheel) without removing them from other groups.

3. **Assign Passwords to users**

[student@localhost ~]$ sudo passwd Supervisor1

[student@localhost ~]$ sudo passwd Engineer1

[student@localhost ~]$ sudo passwd Engineer2

[student@localhost ~]$ sudo passwd Engineer3

[student@localhost ~]$ sudo passwd Operator1

[student@localhost ~]$ sudo passwd Operator2

[student@localhost ~]$ sudo passwd Operator3

Username	Password
Supervisor1	supervisor
Engineer1	engineer
Engineer2	engineer
Engineer3	engineer
Operator1	operator
Operator2	operator
Operator3	operator

EX:

```
[student@localhost ~]$ sudo passwd Supervisor1
Changing password for user Supervisor1.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
```

Then switch to Supervisor1

[student@localhost ~]$ su – Supervisor1

4. Create User Groups

```
[Supervisor1@localhost ~]$ sudo usermod -g Supervisors Supervisor1
[Supervisor1@localhost ~]$ sudo usermod -g Engineers Engineer1
[Supervisor1@localhost ~]$ sudo usermod -g Engineers Engineer2
[Supervisor1@localhost ~]$ sudo usermod -g Engineers Engineer3
[Supervisor1@localhost ~]$ sudo usermod -g Operators Operator1
[Supervisor1@localhost ~]$ sudo usermod -g Operators Operator2
[Supervisor1@localhost ~]$ sudo usermod -g Operators Operator3
[Supervisor1@localhost ~]$ groups Supervisor1
Supervisor1 : Supervisors wheel
```

- **usermod**: Modifies user account properties.
- **-g**: Adds the user to the specified Primary group.
- **Supervisors:** Example group name.
- **Supervisor1:** The user to be added to the group.

5. Create Factory environment in shared place let it be the /home dir.

[Supervisor1@localhost]$ mkdir /home/Factory

6. Create Directories for Tasks and Performance Monitoring

```
[Supervisor1@localhost Tasks]$ mkdir /home/Factory/Tasks/ /home/Factory/Employee_Performance/
```

Create 2 dir (Tasks and Employee_Performance) in the Factory dir using mkdir command.

```
[Supervisor1@localhost home]$ sudo chown Supervisor1:Supervisors Factory/Tasks/ Factory/Employee_Performance/
```

```
[Supervisor1@localhost Tasks]$ sudo chmod 755 /home/Factory/Tasks/
[Supervisor1@localhost Tasks]$ sudo chmod 751 /home/Factory/Employee_Performance/
[Supervisor1@localhost Tasks]$
```

Assign Ownership:

- **chown** : Changes the ownership of a file or directory.
- **user:group** : Assigns the specified user and group as owners.
- **/path/to/directory**: The directory whose ownership is being changed.

Read (r) = 4

Write (w) = 2

Execute (x) = 1

So 7 =4+2+1=rwx

Assign Permissions:

- **chmod :** changes permissions of a file or directory.
- **755 :** gives full permissions to the owner , read (r) and execute (x) to the group and others.
- **/path/to/directory:** The directory whose permissions are being changed.

Note: we at first gave 751 to Tasks dir as permissions so that other users have execute permission to traverse the directories to see their tasks but later on in the complete_task script we needed read permission so we added it.

7. Create Subdirectories for Tasks by Role

```
[Supervisor1@localhost Tasks]$ mkdir /home/Factory/Tasks/Engineers_Tasks/ /home/Factory/Tasks/Operators_Tasks/
```

Create 2 other directories in the Tasks dir (one for Engineers and one for Operators).

```
[Supervisor1@localhost ~]$ chown Supervisor1:Supervisors /home/Factory/Tasks/Engineers_Tasks/
[Supervisor1@localhost ~]$ chown Supervisor1:Supervisors /home/Factory/Tasks/Operators_Tasks/
```

Assign ownership to Supervisor1 and group Supervisors.

```
[Supervisor1@localhost ~]$ sudo chmod 751 /home/Factory/Tasks/Engineers_Tasks/
[Supervisor1@localhost ~]$ sudo chmod 751 /home/Factory/Tasks/Operators_Tasks/
[Supervisor1@localhost ~]$
```

Assigning permissions to be **751:** full access to Supervisor1 , rw to Supervisors group and 1 to others to make it dir traversable using cd assuming the user knows the path from there to his tasks which is inside a dir with his username in this path.

```
[Supervisor1@localhost ~]$ sudo ls -l /home/Factory/Tasks/
total 0
drwxr-x--x. 5 Supervisor1 Supervisors 57 Dec  4 16:22 Engineers_Tasks
drwxr-x--x. 5 Supervisor1 Supervisors 57 Dec  4 16:22 Operators_Tasks
[Supervisor1@localhost ~]$
```

- **`Ls -l`** lists the content while including permissions, ownership, size, and modification date.

Ex: Engineer1 wants to see his tasks in /home/Factory/Tasks/Engineers_Tasks/Engineer1

so, he can either navigate directly to the path or follow this steps:

navigates to /home/Factory from there he can either list the available directories or cd to /Tasks , same goes here in dir /Tasks. Engineer1 is now in /Engineers_Tasks he can't list directories their so he should cd to his dir Engineer1.

8. Create Subdirectories for each user in their corresponding Task dir

```
[Supervisor1@localhost ~]$ for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3; do
> sudo chown $user:Supervisors /home/Factory/Tasks/*/$user
> sudo chmod 570 /home/Factory/Tasks/*/$user
> done
[Supervisor1@localhost ~]$ for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3; do sudo ls -ld /home/Factory/Tasks/*/$us
er; done
dr-xrwx---. 2 Engineer1 Supervisors 177 Dec  4 16:49 /home/Factory/Tasks/Engineers_Tasks/Engineer1
dr-xrwx---. 2 Engineer2 Supervisors 177 Dec  4 16:49 /home/Factory/Tasks/Engineers_Tasks/Engineer2
dr-xrwx---. 2 Engineer3 Supervisors 177 Dec  4 16:49 /home/Factory/Tasks/Engineers_Tasks/Engineer3
dr-xrwx---. 2 Operator1 Supervisors 177 Dec  4 16:50 /home/Factory/Tasks/Operators_Tasks/Operator1
dr-xrwx---. 2 Operator2 Supervisors 177 Dec  4 16:50 /home/Factory/Tasks/Operators_Tasks/Operator2
dr-xrwx---. 2 Operator3 Supervisors 177 Dec  4 16:50 /home/Factory/Tasks/Operators_Tasks/Operator3
[Supervisor1@localhost ~]$
```

Create subdirectories for each user that has tasks using mkdir where every user owns his directory with rw permission, the group owner is Supervisors with full permissions since they will be the ones providing the tasks and with no permissions to others since each dir belongs to its user only.

- Another important thing to be added so that each Engineer can see the corresponding Operator tasks (however not vice versa) is ACL (Access control lists) which modify the permission slightly or in other term add a permission to a specific entity.

```
[Supervisor1@localhost Tasks]$ setfacl -m u:Engineers:x /home/Factory/Tasks/Operators_Tasks/
setfacl: Option -m: Invalid argument near character 3
[Supervisor1@localhost Tasks]$ setfacl -m g:Engineers:x /home/Factory/Tasks/Operators_Tasks/
[Supervisor1@localhost Tasks]$ getfacl /home/Factory/Tasks/Operators_Tasks/
getfacl: Removing leading '/' from absolute path names
# file: home/Factory/Tasks/Operators_Tasks/
# owner: Supervisor1
# group: Supervisors
user::rwx
group::r-x
group:Engineers:--x
mask::r-x
other::--x
```

- **setfacl**: Command to set or modify ACLs for files or directories.
- **-m :** Option to modify ACLs.
- **g:Engineers:x :** Correctly specifies the group (g) Engineers and grants the x (execute) permission.

```
[Supervisor1@localhost Tasks]$ sudo setfacl -m u:Engineer1:rx /home/Factory/Tasks/Operators_Tasks/Operator1
[Supervisor1@localhost Tasks]$ sudo setfacl -m u:Engineer2:rx /home/Factory/Tasks/Operators_Tasks/Operator2
[Supervisor1@localhost Tasks]$ sudo setfacl -m u:Engineer3:rx /home/Factory/Tasks/Operators_Tasks/Operator3
```

- **-m :** Option to modify ACLs.
- **u:<owner_user>:x :** Correctly specifies the user (u) (Ex :Engineer1) and grants the rx (read and execute) permission to be able to list the content of the corresponding Operator.

```
[Supervisor1@localhost Tasks]$ getfacl /home/Factory/Tasks/Operators_Tasks/Operator1
getfacl: Removing leading '/' from absolute path names
# file: home/Factory/Tasks/Operators_Tasks/Operator1
# owner: Operator1
# group: Supervisors
user::r-x
user:Engineer1:r-x
group::rwx
mask::rwx
other::---

[Supervisor1@localhost Tasks]$ getfacl /home/Factory/Tasks/Operators_Tasks/Operator2
getfacl: Removing leading '/' from absolute path names
# file: home/Factory/Tasks/Operators_Tasks/Operator2
# owner: Operator2
# group: Supervisors
user::r-x
user:Engineer2:r-x
group::rwx
mask::rwx
other::---

[Supervisor1@localhost Tasks]$ getfacl /home/Factory/Tasks/Operators_Tasks/Operator3
getfacl: Removing leading '/' from absolute path names
# file: home/Factory/Tasks/Operators_Tasks/Operator3
# owner: Operator3
# group: Supervisors
user::r-x
user:Engineer3:r-x
group::rwx
mask::rwx
other::---
```

- **getfacl :** Command to retrieve and display the ACLs of a file or directory.
- **/home/Factory/Tasks/Operators_Tasks/Operator1:** Path to the directory whose ACLs are being displayed.

9. Creating Task Files

```
[Supervisor1@localhost home]$ for i in {1..10};do
> touch Factory/Tasks/Engineers_Tasks/Engineer1/Task$i.txt
> touch Factory/Tasks/Engineers_Tasks/Engineer2/Task$i.txt
> touch Factory/Tasks/Engineers_Tasks/Engineer3/Task$i.txt
> done
[Supervisor1@localhost home]$ for i in {1..10};do
> touch Factory/Tasks/Operators_Tasks/Operator1/Task$i.txt
> touch Factory/Tasks/Operators_Tasks/Operator2/Task$i.txt
> touch Factory/Tasks/Operators_Tasks/Operator3/Task$i.txt
> done
[Supervisor1@localhost home]$
```

Create 10 tasks for every user using touch.

```
[Supervisor1@localhost home]$ for i in 1 2 3;do
> ls Factory/Tasks/Engineers_Tasks/Engineer$i
> ls Factory/Tasks/Operators_Tasks/Operator$i
> done
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
Task10.txt  Task1.txt  Task2.txt  Task3.txt  Task4.txt  Task5.txt  Task6.txt  Task7.txt  Task8.txt  Task9.txt
[Supervisor1@localhost home]$
```

```
[Supervisor1@localhost ~]$ for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3;
> do
> sudo chown $user:Supervisors /home/Factory/Tasks/*/$user/Task{1..10}.txt
> done
[Supervisor1@localhost ~]$
```

```
[Supervisor1@localhost ~]$ sudo chmod 570 /home/Factory/Tasks/Engineers_Tasks/Engineer{1..3}/Task{1..10}.txt
[Supervisor1@localhost ~]$ sudo chmod 570 /home/Factory/Tasks/Operators_Tasks/Operator{1..3}/Task{1..10}.txt
```

Each user should own his tasks and the group be Supervisors.

Setting the permission for every task to be 570 giving the owner rw permissions, full permissions to group Supervisors and no access to others.

```
[Supervisor1@localhost ~]$ ls -l /home/Factory/Tasks/Engineers_Tasks/Engineer3
total 0
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task10.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task1.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task2.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task3.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task4.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task5.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task6.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task7.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task8.txt
-r-xrwx---. 1 Engineer3 Supervisors 0 Dec  4 16:49 Task9.txt
```

```
[Supervisor1@localhost ~]$ ls -l /home/Factory/Tasks/Operators_Tasks/Operator1
total 0
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task10.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task1.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task2.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task3.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task4.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task5.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task6.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task7.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task8.txt
-r-xrwx---. 1 Operator1 Supervisors 0 Dec  4 16:50 Task9.txt
```

List the long listing to make sure of ownership and permissions.

10. Create Directory for Completed Tasks

```
[Supervisor1@localhost home]$ mkdir Factory/Employee_Performance/Completed_Tasks
[Supervisor1@localhost home]$ for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3;do
> mkdir Factory/Employee_Performance/Completed_Tasks/$user
> done
[Supervisor1@localhost home]$ ls Factory/Employee_Performance/Completed_Tasks/
Engineer1  Engineer2  Engineer3  Operator1  Operator2  Operator3
[Supervisor1@localhost home]$
```

Create a dir for Completed tasks in Employee_Performance then create subdirectories for each user.

```
[Supervisor1@localhost Factory]$ chown Supervisor1:Supervisors /home/Factory/Employee_Performance/Completed_Tasks/
[Supervisor1@localhost Factory]$ chmod 751 /home/Factory/Employee_Performance/Completed_Tasks/
[Supervisor1@localhost Factory]$
```

Set the ownership using chown and permissions to be 751.

```
[Supervisor1@localhost Factory]$ for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3;do
> sudo chown $user:Supervisors /home/Factory/Employee_Performance/Completed_Tasks/$user
> sudo chmod 770 /home/Factory/Employee_Performance/Completed_Tasks/$user
> done
[sudo] password for Supervisor1:
[Supervisor1@localhost Factory]$ ls -l /home/Factory/Employee_Performance/Completed_Tasks/
total 0
drwxrwx---. 2 Engineer1 Supervisors 6 Dec  4 16:54 Engineer1
drwxrwx---. 2 Engineer2 Supervisors 6 Dec  4 16:54 Engineer2
drwxrwx---. 2 Engineer3 Supervisors 6 Dec  4 16:54 Engineer3
drwxrwx---. 2 Operator1 Supervisors 6 Dec  4 16:54 Operator1
drwxrwx---. 2 Operator2 Supervisors 6 Dec  4 16:54 Operator2
drwxrwx---. 2 Operator3 Supervisors 6 Dec  4 16:54 Operator3
[Supervisor1@localhost Factory]$
```

For each subdirectory it is owned by the user himself and the group is Supervisors same as the sub directories in /Engineers_Tasks and /Operators_Tasks. However, the permissions are the slightly different (770) as the user need write permission to copy the task using the script from his directory in /Engineers_Tasks/Engineer1 to /Completed_Tasks/Engineer1.

11. Bash Script for Completing Tasks

```
[Supervisor1@localhost home]$ sudo vim Factory/complete_task.sh
[sudo] password for Supervisor1:
```

Create the script and edit it using vim

```bash
#!/bin/bash

#check if user provided the right args
if [ -z "$1" ];then
        echo -e "\033[31mUsage:./complete_task.sh <task_file>.\033[0m"
        exit 1
fi

user=$(whoami)
dest="/home/Factory/Employee_Performance/Completed_Tasks/${user}"

#check if the task is already completed
if [ -f "$dest"/"$1" ];then
        echo -e "\033[33m"$1" has already been completed!\033[0m"
        exit 0
fi

#check if the file exists
if [ -f /home/Factory/Tasks/*/${user}/"$1" ];then
        cp /home/Factory/Tasks/*/"$user"/"$1" "$dest"
        echo -e "\033[32m $1 has been successfully marked as completed and moved to Completed_Tasks/"$user".\033[0m"
else
        echo -e "\033[31mError: Task not found.\033[0m"
        exit 1
fi
```

- **[-z "$1"] checks if the first argument ($1) is empty.(no task provided)**
    - **If no prints usage message in red (\033[31m) and exit with code 1 (Error).**

- **Get the acting user using (whoami).**
- **Get the destination path and save it in dest.**

- **Check if the task has been completed (I.e already in dest) [-f "$dest"/"$1"]**
    - **If yes print message in yellow (\033[33m) and exit with code 0.**

- **Check if the task file is in source dir /**
    - **If yes then copy to dest and print an echo message.**
    - **If no print an error in red (\033[32m) and exit with code 1.**

12. Bash Script for KPI Calculation

```
[Supervisor1@localhost home]$ sudo vim Factory/get_kpi.sh
[Supervisor1@localhost home]$
```

Create the script and edit it using vim

```
#!/bin/bash


for user in Engineer1 Engineer2 Engineer3 Operator1 Operator2 Operator3;do

        completed=$(ls /home/Factory/Employee_Performance/Completed_Tasks/"$user" | wc -l)
        KPI=$(echo "scale=2;$completed/10" | bc)

        file="/home/Factory/Employee_Performance/${user}_KPI.txt"
        timestamp=$(date '+%Y-%m-%d %H:%M:%S')
        echo "KPI; $KPI; Timestamp: $timestamp" >> "$file"
done
~
```

KPI =no.completed Tasks / total Tasks

- **wc -l :Cont lines from the ouput of the ls using a pipe (|) in between (i.e count how many tasks are completed).**

- **scale=2 : 2 decimal places for the result.**
- **bc: A basic calculator used to handle floating-point arithmetic.**

- **generate the destination of the user_KPI.txt.**
- **get the timestamp.**

- **print the KPI along with the time stamp in the created destination (file)**

13. Scripts Ownership and permissions

```
[Supervisor1@localhost ~]$ sudo chown Supervisor1:Supervisors /home/Factory/complete_task.sh
[Supervisor1@localhost ~]$ sudo chown Supervisor1:Supervisors /home/Factory/get_kpi.sh
[Supervisor1@localhost ~]$
```

Set ownership to Supervisor1: Supervisors(group).

```
[Supervisor1@localhost Factory]$ chmod +x complete_task.sh
[Supervisor1@localhost Factory]$ chmod u+x get_kpi.sh
[Supervisor1@localhost Factory]$ chmod g+x get_kpi.sh
[Supervisor1@localhost Factory]$ ls -l
total 8
-rwxr-xr-x. 1 Supervisor1 Supervisors 330 Dec  4 19:34 complete_task.sh
drwxr-x--x. 3 Supervisor1 Supervisors  29 Dec  4 16:53 Employee_Performance
-rwxr-xr--. 1 Supervisor1 Supervisors 306 Dec  4 19:35 get_kpi.sh
drwxr-x--x. 4 Supervisor1 Supervisors  52 Dec  4 16:22 Tasks
[Supervisor1@localhost Factory]$
```

- Make the complete_task script executable (chmod +x add execute to all roles)
- Make the get_kpi script executable for user (u+x) and group (g+x) only since they are the supervisors.

Testing Permissions:

```
[Engineer1@localhost home]$ cd Factory/
[Engineer1@localhost Factory]$ ls
complete_task.sh  Employee_Performance  get_kpi.sh  Tasks
[Engineer1@localhost Factory]$ cd Tasks/
[Engineer1@localhost Tasks]$ cd Engineers_Tasks
[Engineer1@localhost Engineers_Tasks]$ ls
ls: cannot open directory '.': Permission denied
[Engineer1@localhost Engineers_Tasks]$ cd Engineer1
[Engineer1@localhost Engineer1]$ ls
Task10.txt  Task2.txt  Task4.txt  Task6.txt  Task8.txt
Task1.txt   Task3.txt  Task5.txt  Task7.txt  Task9.txt
[Engineer1@localhost Engineer1]$
```

In this screenshot:

- **Engineer1 can traverse the directories while also able to view his tasks but unable to view other dirs in Engineers_Tasks**

```
[Engineer1@localhost Completed_Tasks]$ ls
Engineer1  Engineer2  Engineer3  Operator1  Operator2  Operator3
[Engineer1@localhost Completed_Tasks]$ cd Engineer2
bash: cd: Engineer2: Permission denied
[Engineer1@localhost Completed_Tasks]$ cd Engineer1
[Engineer1@localhost Engineer1]$ ls
[Engineer1@localhost Engineer1]$ cd ../Operator1
```

- **Engineer1 cannot access Engineer2 Completed tasks.**

```
[Engineer1@localhost Tasks]$ cd Operators_Tasks
[Engineer1@localhost Operators_Tasks]$ ls
ls: cannot open directory '.': Permission denied
[Engineer1@localhost Operators_Tasks]$ cd Operator1
[Engineer1@localhost Operator1]$ ls
Task10.txt  Task2.txt  Task4.txt  Task6.txt  Task8.txt
Task1.txt   Task3.txt  Task5.txt  Task7.txt  Task9.txt
[Engineer1@localhost Operator1]$
```

- **Engineer1 cannot access or view any operator Tasks except those of Operator1**

```
[Engineer1@localhost Operator1]$ cd ../Operator2
bash: cd: ../Operator2: Permission denied
[Engineer1@localhost Operator1]$
```

- **As shown, he cannot Access Operator2 dir.**

```
[Operator1@localhost Factory]$ cd Tasks/Engineers_Tasks
[Operator1@localhost Engineers_Tasks]$ ls
ls: cannot open directory '.': Permission denied
[Operator1@localhost Engineers_Tasks]$ cd Engineer1
-bash: cd: Engineer1: Permission denied
[Operator1@localhost Engineers_Tasks]$
```

- **Same goes for any operator and engineer. each can access his specified dir but not any other of user except in case of engineers viewing the corresponding operators.**
- **As shown her Operator1 tried to access Engineer1 dir to see his tasks but he couldn't.**

Scripts Testing:

Let the Engineer1 submit Task1 and 2 and Operator1 submit Task1, 5 and 10.

```
[Engineer1@localhost ~]$ cd /home/Factory/
[Engineer1@localhost Factory]$ ./complete_task.sh
Usage:./complete_task.sh <task_file>.
[Engineer1@localhost Factory]$ ./complete_task.sh Tasks11.txt
Error: Task not found.
[Engineer1@localhost Factory]$ ./complete_task.sh Tasks1.txt
Error: Task not found.
[Engineer1@localhost Factory]$ ./complete_task.sh Task1.txt
Task1.txt has been successfully marked as completed and moved to Completed_Tasks/Engineer1.
[Engineer1@localhost Factory]$ ./complete_task.sh Task2.txt
Task2.txt has been successfully marked as completed and moved to Completed_Tasks/Engineer1.
[Engineer1@localhost Factory]$ ./complete_task.sh Task1.txt
Task1.txthas already been completed!
[Engineer1@localhost Factory]$ ./complete_task.sh Task2.txt
Task2.txthas already been completed!
```

```
[Engineer1@localhost Factory]$ su - Operator1
Password:
[Operator1@localhost ~]$ cd /home/Factory/
[Operator1@localhost Factory]$ ./complete_task.sh Task1.txt
Task1.txt has been successfully marked as completed and moved to Completed_Tasks/Operator1.
[Operator1@localhost Factory]$ ./complete_task.sh Task5.txt
Task5.txt has been successfully marked as completed and moved to Completed_Tasks/Operator1.
[Operator1@localhost Factory]$ ./complete_task.sh Task10.txt
Task10.txt has been successfully marked as completed and moved to Completed_Tasks/Operator1.
[Operator1@localhost Factory]$ ./complete_task.sh Task10.txt
Task10.txthas already been completed!
```

Login with supervisor to get the kpi and check the completed tasks.

```
[Operator1@localhost Factory]$ su - Supervisor1
Password:
[Supervisor1@localhost ~]$ cd /home/Factory/
[Supervisor1@localhost Factory]$ ./get_kpi.sh
```

```
[Supervisor1@localhost Factory]$ ls -l Employee_Performance/Completed_Tasks/Engineer1
total 0
-r-xr-x---. 1 Engineer1 Engineers 0 Dec  4 23:58 Task1.txt
-r-xr-x---. 1 Engineer1 Engineers 0 Dec  4 23:58 Task2.txt
[Supervisor1@localhost Factory]$ ls -l Employee_Performance/Completed_Tasks/Operator1
total 0
-r-xr-x---. 1 Operator1 Operators 0 Dec  4 23:59 Task10.txt
-r-xr-x---. 1 Operator1 Operators 0 Dec  4 23:59 Task1.txt
-r-xr-x---. 1 Operator1 Operators 0 Dec  4 23:59 Task5.txt
```

```
[Supervisor1@localhost Factory]$ ls -l Employee_Performance/
total 24
drwxr-x--x. 8 Supervisor1 Supervisors 108 Dec  4 16:54 Completed_Tasks
-rw-r--r--. 1 Supervisor1 Supervisors  41 Dec  5 00:00 Engineer1_KPI.txt
-rw-r--r--. 1 Supervisor1 Supervisors  39 Dec  5 00:00 Engineer2_KPI.txt
-rw-r--r--. 1 Supervisor1 Supervisors  39 Dec  5 00:00 Engineer3_KPI.txt
-rw-r--r--. 1 Supervisor1 Supervisors  41 Dec  5 00:00 Operator1_KPI.txt
-rw-r--r--. 1 Supervisor1 Supervisors  39 Dec  5 00:00 Operator2_KPI.txt
-rw-r--r--. 1 Supervisor1 Supervisors  39 Dec  5 00:00 Operator3_KPI.txt
[Supervisor1@localhost Factory]$ cat Employee_Performance/Engineer1_KPI.txt
KPI; .20; Timestamp: 2024-12-05 00:00:37
[Supervisor1@localhost Factory]$ cat Employee_Performance/Operator1_KPI.txt
KPI; .30; Timestamp: 2024-12-05 00:00:37
[Supervisor1@localhost Factory]$
```

Let operator1 submit another task (Task6). when the Supervisor calculates the kpi again it is appended at the end of the Operator1_KPI.txt with the timestamp.

```
[Operator1@localhost Factory]$ ./complete_task.sh Task6.txt
Task6.txt has been successfully marked as completed and moved to Completed_Tasks/Operator1.
[Operator1@localhost Factory]$ su - Supervisor1
Password:
[Supervisor1@localhost ~]$ cd /home/Factory/
[Supervisor1@localhost Factory]$ ./get_kpi.sh
[Supervisor1@localhost Factory]$ ls Employee_Performance/Operator1_KPI.txt
Employee_Performance/Operator1_KPI.txt
[Supervisor1@localhost Factory]$ cat Employee_Performance/Operator1_KPI.txt
KPI; .30; Timestamp: 2024-12-05 00:00:37
KPI; .40; Timestamp: 2024-12-05 00:18:20
[Supervisor1@localhost Factory]$
```