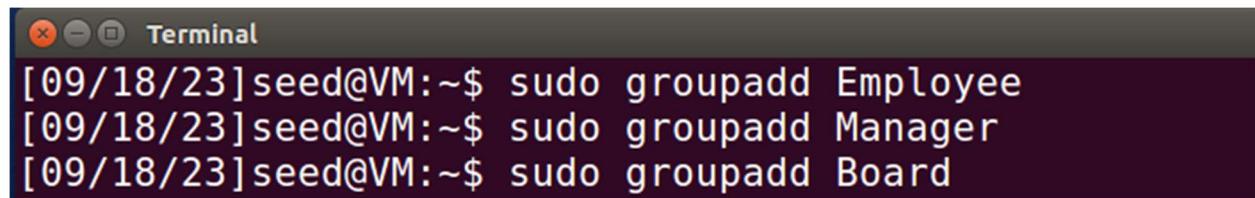


Task 1: Add Groups and Users

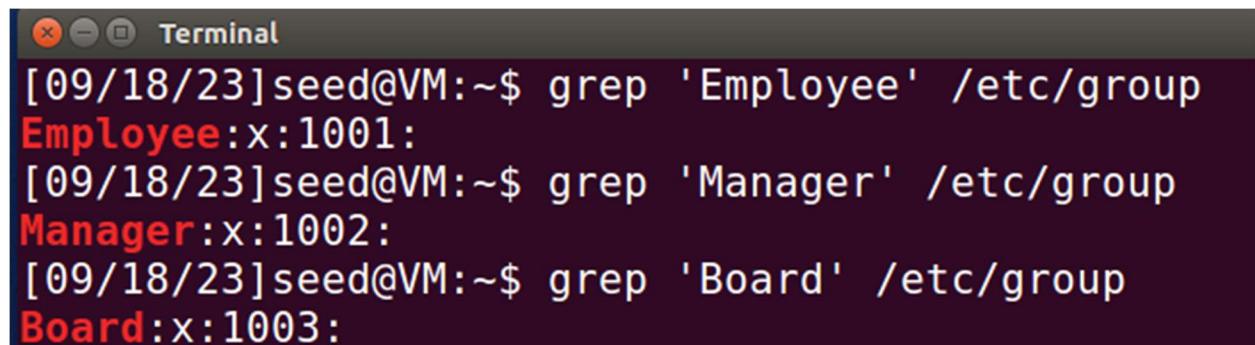
1. Making the three groups



```
[09/18/23] seed@VM:~$ sudo groupadd Employee  
[09/18/23] seed@VM:~$ sudo groupadd Manager  
[09/18/23] seed@VM:~$ sudo groupadd Board
```

A screenshot of a terminal window titled "Terminal". The window has a dark background and light-colored text. It shows three commands being run sequentially: "sudo groupadd Employee", "sudo groupadd Manager", and "sudo groupadd Board".

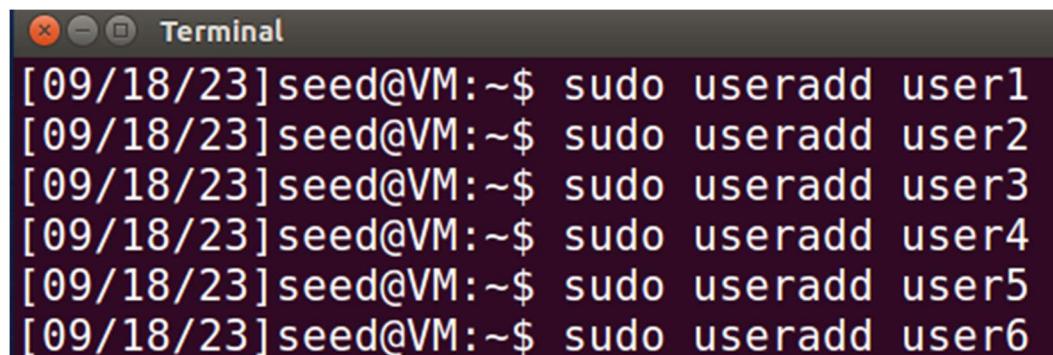
Verify groups were made successfully:



```
[09/18/23] seed@VM:~$ grep 'Employee' /etc/group  
Employee:x:1001:  
[09/18/23] seed@VM:~$ grep 'Manager' /etc/group  
Manager:x:1002:  
[09/18/23] seed@VM:~$ grep 'Board' /etc/group  
Board:x:1003:
```

A screenshot of a terminal window titled "Terminal". It displays the results of running "grep" against the "/etc/group" file to search for the names of the three groups: "Employee", "Manager", and "Board". The output shows each group name followed by its GID (1001, 1002, and 1003 respectively) and a colon.

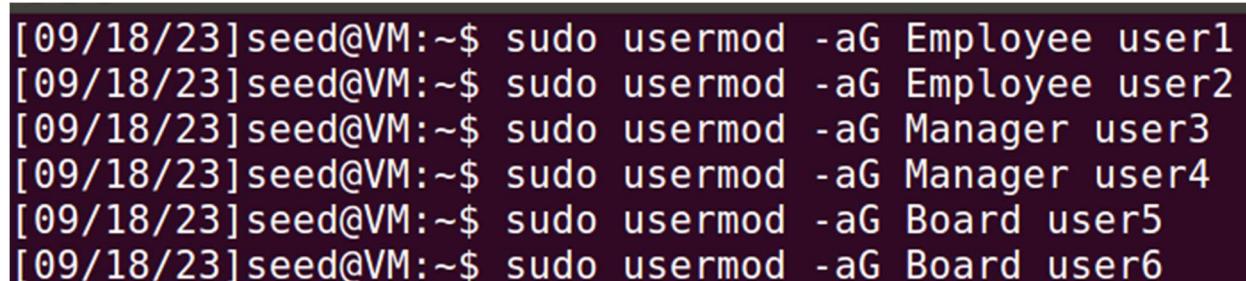
2. Add users



```
[09/18/23] seed@VM:~$ sudo useradd user1  
[09/18/23] seed@VM:~$ sudo useradd user2  
[09/18/23] seed@VM:~$ sudo useradd user3  
[09/18/23] seed@VM:~$ sudo useradd user4  
[09/18/23] seed@VM:~$ sudo useradd user5  
[09/18/23] seed@VM:~$ sudo useradd user6
```

A screenshot of a terminal window titled "Terminal". It shows six commands using "sudo useradd" to create six new user accounts: "user1" through "user6".

Assign users to groups, two users in each group and each user in only one group each:



```
[09/18/23] seed@VM:~$ sudo usermod -aG Employee user1  
[09/18/23] seed@VM:~$ sudo usermod -aG Employee user2  
[09/18/23] seed@VM:~$ sudo usermod -aG Manager user3  
[09/18/23] seed@VM:~$ sudo usermod -aG Manager user4  
[09/18/23] seed@VM:~$ sudo usermod -aG Board user5  
[09/18/23] seed@VM:~$ sudo usermod -aG Board user6
```

A screenshot of a terminal window titled "Terminal". It shows six commands using "sudo usermod" with the "-aG" option to add each user to their respective group. User "user1" and "user2" are added to the "Employee" group, users "user3" and "user4" are added to the "Manager" group, and users "user5" and "user6" are added to the "Board" group.

Verify that the users were successfully added to the groups:

```
[09/18/23]seed@VM:~$ groups user1
user1 : user1 Employee
[09/18/23]seed@VM:~$ groups user2
user2 : user2 Employee
[09/18/23]seed@VM:~$ groups user3
user3 : user3 Manager
[09/18/23]seed@VM:~$ groups user4
user4 : user4 Manager
[09/18/23]seed@VM:~$ groups user5
user5 : user5 Board
[09/18/23]seed@VM:~$ groups user6
user6 : user6 Board
```

Employee: user1 and user2

Manager: user3 and user4

Board: user5 and user6

Task 2: Role Based Access Control

1. File Design:

Employee Files: These files are related to general information, common for every employee. Example created by user1

Files: “employee_info.txt” and “employee_info2.txt”

Manager Files: These files are related to managerial tasks, like scheduling or budgets. Example created by user3.

Files: “manager_task1.txt” and “manager_task2.txt”

Board Files: These files containing sensitive company information or long-term plans. Example created by user5.

Files: “board_plan1.txt” and “board_plan2.txt”

Access Rights Setting:

Employee Files: Here is an example of how to ensure that the owner user1 can read, write, and execute while the group “Employee” can only read the files. Others cannot access. So user2 can read the file, but not write or execute the file. Similarly, user3-6 cannot read, write, or execute the file.

```
sudo chown user1:Employee employee_info*.txt
```

```
sudo chmod 740 employee_info*.txt
```

Manager Files: Here is an example of how to ensure that owner user3 can read, write, and execute while the group “Manager” can only read the files. Others cannot access. So user4 can read the file, but not write or execute the file. Similarly, user1-2 and user5-6 cannot read, write, or execute the file.

```
sudo chown user3:Manager manager_task*.txt
```

```
sudo chmod 740 manager_task*.txt
```

Board Files: here is an example of how to ensure that owner user5 can read, write, and execute while the group “Board” can only read the files. Others cannot access. So user6 can read the file, but not write or execute the file. Similarly, user1-4 cannot read, write, or execute the file.

```
sudo chown user5:Board board_plan*.txt
```

```
sudo chmod 740 board_plan*.txt
```

2. The root user in Linux, also known as the superuser, has unrestricted access to all files and commands. It is not possible to prevent root from accessing any files, as root inherently has the permission to override or change permissions. This highlights how important it is that root access is restricted, because if compromised, the whole system is vulnerable.

For the seed user (if it’s not a root user or equivalent), you can use similar permissions as discussed above to restrict access. However, if seed has sudo privileges, it can override these permissions to still access the files. Without sudo privileges, seed can be restricted.

3. Implementation and Testing:

Using the above permission examples and files, I will carry out some testing with the users. First is a screenshot showing the example files outlined above.

user1, user3, and user5 will be tested as the owner of their respective files.

user2, user4, and user6 will be tested as the group but not owner of the files for their respective groups as shown previously.

File setup:

```
[09/18/23]seed@VM:~/lab2$ ls -l
total 24
-rwxr----- 1 user5 Board      48 Sep 18 14:34 board_plan1.txt
-rwxr----- 1 user5 Board      39 Sep 18 14:34 board_plan2.txt
-rwxr----- 1 user1 Employee   46 Sep 18 14:32 employee_info1.txt
-rwxr----- 1 user1 Employee   44 Sep 18 14:32 employee_info2.txt
-rwxr----- 1 user3 Manager    49 Sep 18 14:33 manager_task1.txt
-rwxr----- 1 user3 Manager    47 Sep 18 14:33 manager_task2.txt
```

Employee Group Test:

user1 access to employee_info1.txt, shows access to file and the user is able to write to the file:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat employee_infol.txt  
This is a file for user1 in the group Employee[09/18/23]seed@VM:~/lab2$ sudo -u  
user1 vim employee_infol.txt  
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat employee_infol.txt  
This is a file for user1 in the group Employee, edit  
[09/18/23]seed@VM:~/lab2$
```

user2, in the Employee group, is able to read the file but not write or execute:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat employee_info1.txt  
This is a file for user1 in the group Employee, edit
```

```
[09/18/23]seed@VM:~/lab2$ sudo -u user2 vim employee info1.txt
```

user3-6, not in Employee group and not the owner, are unable to read, write, or execute the file:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user3 cat employee_inf01.txt  
cat: employee_inf01.txt: Permission denied  
[09/18/23]seed@VM:~/lab2$ sudo -u user4 cat employee_inf01.txt  
cat: employee_inf01.txt: Permission denied  
[09/18/23]seed@VM:~/lab2$ sudo -u user5 cat employee_inf01.txt  
cat: employee_inf01.txt: Permission denied  
[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat employee_inf01.txt  
cat: employee_inf01.txt: Permission denied
```

Manager Group Test:

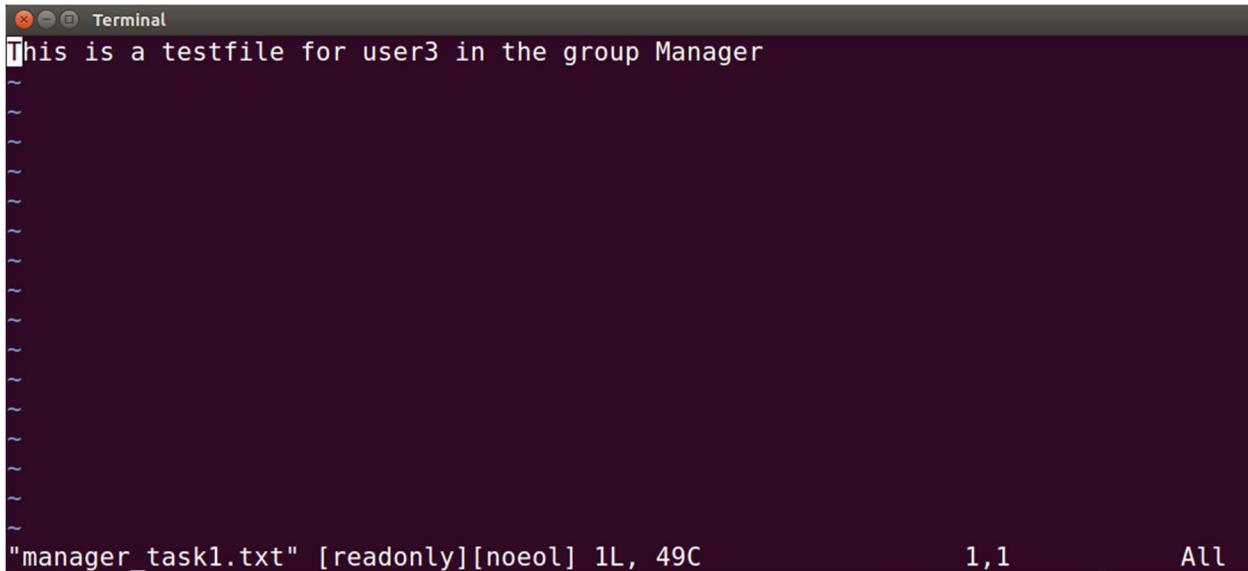
user3 access to manager_task1.txt, shows access to file and the user is able to write to the file:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user3 cat manager_task1.txt
This is a testfile for user3 in the group Manager[09/18/23]seed@VM:~/lab2$ sudo
-u user3 vim manager_task1.txt
[09/18/23]seed@VM:~/lab2$
```

user4, in the Manager group, is able to read the file but not write or execute:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user4 cat manager_task1.txt  
This is a testfile for user3 in the group Manager[09/18/23]seed@VM:~/lab2$ █
```

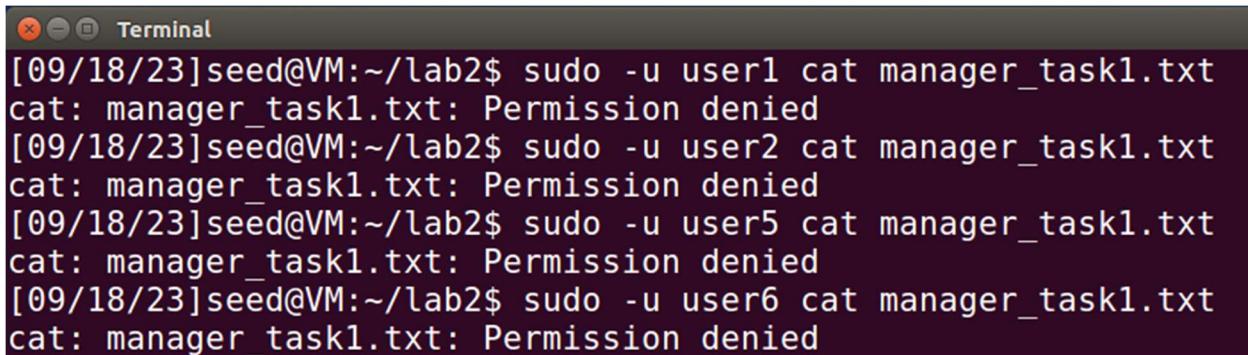
```
[09/18/23]seed@VM:~/lab2$ sudo -u user5 vim board_plan1.txt  
[09/18/23]seed@VM:~/lab2$
```



This is a testfile for user3 in the group Manager

"manager_task1.txt" [readonly][noeol] 1L, 49C 1,1 All

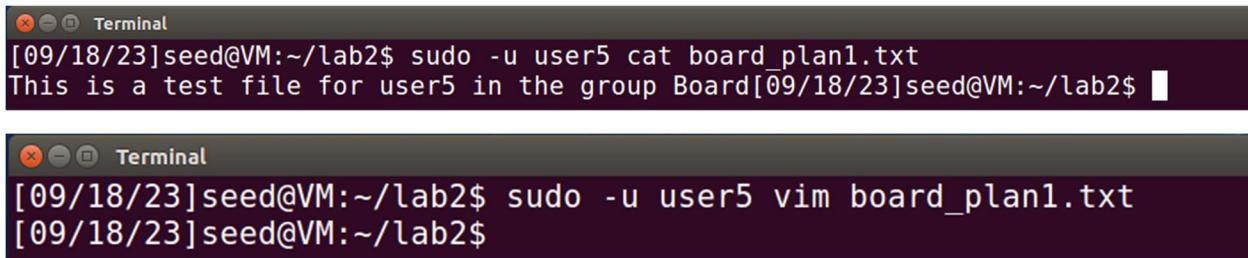
user1-2 and user5-6, not in Manager group and not the owner, are unable to read, write, or execute the file:



```
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat manager_task1.txt
cat: manager_task1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat manager_task1.txt
cat: manager_task1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user5 cat manager_task1.txt
cat: manager_task1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat manager_task1.txt
cat: manager_task1.txt: Permission denied
```

Board Group Test:

user5 access to board_plan1.txt, shows access to file and the user is able to write to the file:



```
[09/18/23]seed@VM:~/lab2$ sudo -u user5 cat board_plan1.txt
This is a test file for user5 in the group Board[09/18/23]seed@VM:~/lab2$ █
```

```
[09/18/23]seed@VM:~/lab2$ sudo -u user5 vim board_plan1.txt
[09/18/23]seed@VM:~/lab2$ █
```

user6, in the Board group, is able to read the file but not write or execute:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat board_plan1.txt  
This is a test file for user5 in the group Board[09/18/23]seed@VM:~/lab2$ █
```

```
[09/18/23]seed@VM:~/lab2$ sudo -u user6 vim board_plan1.txt  
[09/18/23]seed@VM:~/lab2$ █
```

user1-4, not in Board group and not the owner, are unable to read, write, or execute the file:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat board_plan1.txt
cat: board_plan1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat board_plan1.txt
cat: board_plan1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user3 cat board_plan1.txt
cat: board_plan1.txt: Permission denied
[09/18/23]seed@VM:~/lab2$ sudo -u user4 cat board_plan1.txt
cat: board_plan1.txt: Permission denied
```

Root is able to access files, because root has unrestricted access:

```
root@VM:/home/seed/lab2
root@VM:/home/seed/lab2# cat employee_info1.txt
This is a file for user1 in the group Employee, edit
root@VM:/home/seed/lab2# cat manager_task1.txt
This is a testfile for user3 in the group Manager

root@VM:/home/seed/lab2# cat board_plan1.txt
This is a test file for user5 in the group Board
```

seed is able to access files using sudo in this example, if sudo is not possible, then seed would not work:

```
[09/18/23]seed@VM:~/lab2$ sudo cat employee_info1.txt
This is a file for user1 in the group Employee, edit
[09/18/23]seed@VM:~/lab2$ sudo cat manager_task1.txt
This is a testfile for user3 in the group Manager
[09/18/23]seed@VM:~/lab2$ sudo cat board_plan1.txt
This is a test file for user5 in the group Board
[09/18/23]seed@VM:~/lab2$
```

Verify File Ownership and Permissions:

```
[09/18/23]seed@VM:~/lab2$ ls -l employee_info1.txt
-rw-r----- 1 user1 Employee 53 Sep 18 14:37 employee_info1.txt
[09/18/23]seed@VM:~/lab2$ ls -l manager_task1.txt
-rw-r----- 1 user3 Manager 49 Sep 18 14:33 manager_task1.txt
[09/18/23]seed@VM:~/lab2$ ls -l board_plan1.txt
-rw-r----- 1 user5 Board 48 Sep 18 14:34 board_plan1.txt
[09/18/23]seed@VM:~/lab2$
```

In these test scenarios it shows that the RBAC design and implementation works as intended.

Task 3: Multi-level Security

1. Hierarchy: Board > Manager > Employee

File Design:

Employee Files: Previously created by user1

Files: "employee_info1.txt" and "employee_info2.txt"

Manager Files: Previously created by user3

Files: "manager_task1.txt" and "manager_task2.txt"

Board Files: Previously created by user5

Files: "board_plan1.txt" and "board_plan2.txt"

In order to accomplish this in stock Linux, since stock Linux doesn't natively support a hierarchical or multi-level access control system readily, I will be using ACL to make this possible.

```
sudo apt install acl
```

Employee Group Permissions:

- Owner: Read, write, execute
- Group Member: Full access rights to any file in Employee Group

Manager Group Permissions:

- Owner: Read, write, execute
- Group Member: Full access rights to any file in Employee and Manager Groups

Board Group Permissions:

- Owner: Read, write, execute
- Group Member: Full access rights to any file in Employee, Manager, and Board Groups

Access Rights Setting:

Employee Files:

```
sudo chown user1:Employee employee_info*.txt  
sudo chmod 770 employee_info*.txt  
sudo setfacl -m g:Employee:rwx employee_info*.txt  
sudo setfacl -m g:Manager:rwx employee_info*.txt
```

```
sudo setfacl -m g:Board:rwx employee_info*.txt
```

This ensures that group members of Board, Manager, and Employee have full access rights for the given Employee group files. All users will have full access rights on these files.

Manager Files:

```
sudo chown user3:Manager manager_task*.txt
```

```
sudo chmod 770 manager_task*.txt
```

```
sudo setfacl -m g:Manager:rwx manager_task*.txt
```

```
sudo setfacl -m g:Board:rwx manager_task *.txt
```

This ensures that group members of Board and Manager have full access rights for the given Manager group files. The following users will have full access rights on these files: user3, user4, user5, and user6.

Board Files:

```
sudo chown user5:Board board_plan*.txt
```

```
sudo chmod 770 board_plan*.txt
```

```
sudo setfacl -m g:Board:rwx board_plan*.txt
```

This ensures that the group members of Board have full access rights for the given Board group files. The following users will have full access rights on these files: user5 and user6.

Even though this method is functional and accomplishes the goal of the multi-level security design, there are still some obstacles to overcome. The obstacles that are encountered here is that while this works, it can get very complicated if there are many files and roles. It's not as intuitive or easy to manage as a true multi-level security model. Since it is more complicated as well, ensuring that the proper permissions are established is crucial, otherwise there can be more vulnerabilities and lapses in security. This may work in a situation where there are few roles and files, but scaling this process up for a large number of files and roles isn't feasible.

2. Access Checking and Testing:**File Setup:**

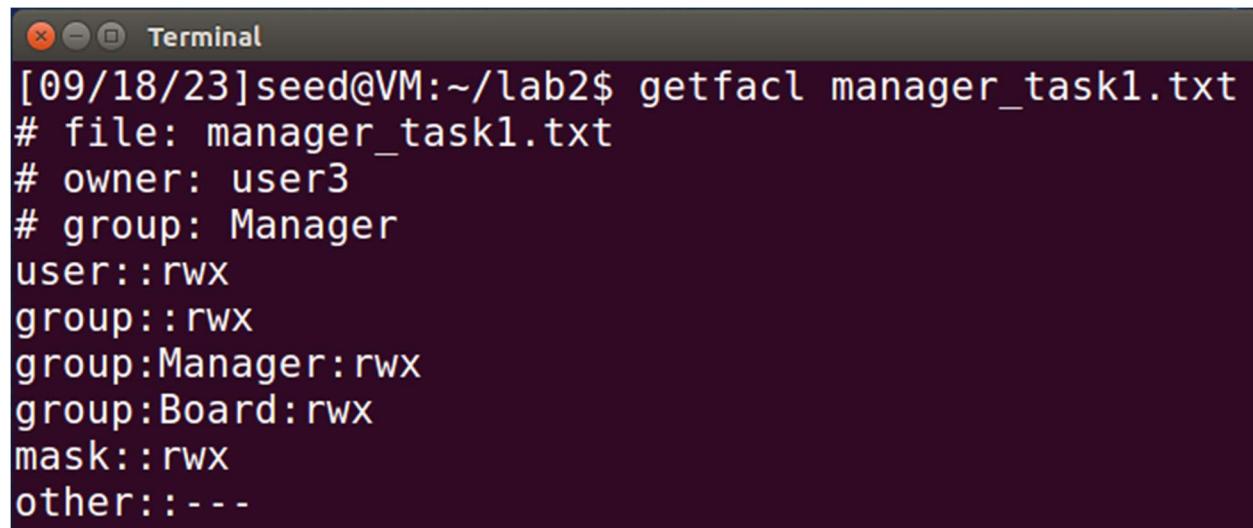
```
Terminal
[09/18/23]seed@VM:~/lab2$ ls -l
total 24
-rwxrwx---+ 1 user5 Board      48 Sep 18 14:34 board_plan1.txt
-rwxrwx---+ 1 user5 Board      39 Sep 18 14:34 board_plan2.txt
-rwxrwx---+ 1 user1 Employee   53 Sep 18 14:37 employee_info1.txt
-rwxrwx---+ 1 user1 Employee   44 Sep 18 14:32 employee_info2.txt
-rwxrwx---+ 1 user3 Manager    49 Sep 18 14:33 manager_task1.txt
-rwxrwx---+ 1 user3 Manager    47 Sep 18 14:33 manager_task2.txt
```

ACL settings for an example file from each group:

All employee_info*.txt files should look like this:

```
Terminal
[09/18/23]seed@VM:~/lab2$ getfacl employee_info1.txt
# file: employee_info1.txt
# owner: user1
# group: Employee
user::rwx
group::---
group:Employee:rwx
group:Manager:rwx
group:Board:rwx
mask::rwx
other::---
```

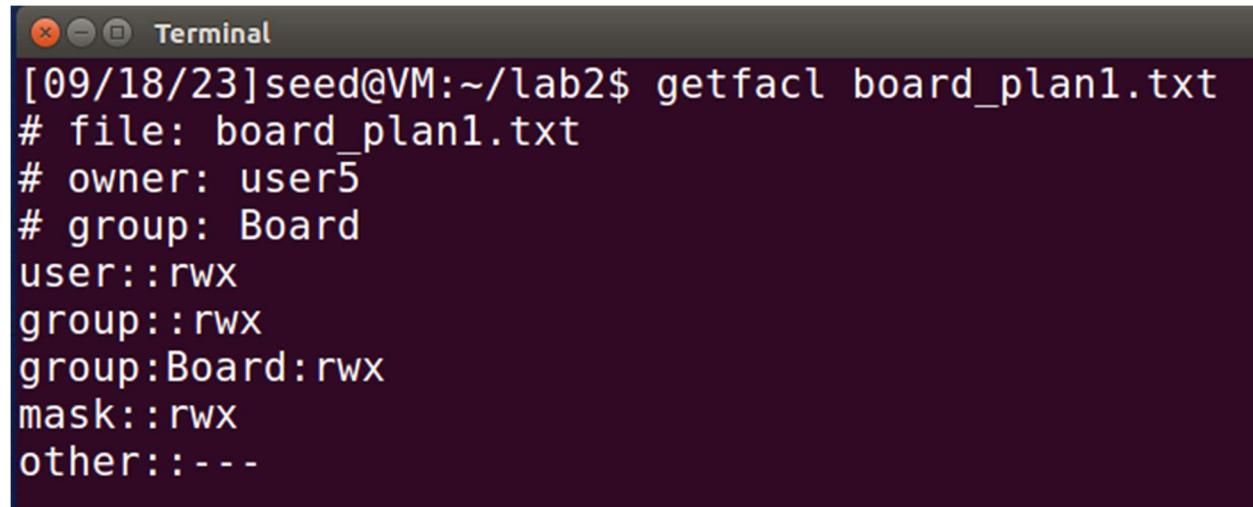
All manager_task*.txt files should look like this:



A screenshot of a terminal window titled "Terminal". The command entered is "getfacl manager_task1.txt". The output shows the file's access control list (ACL) with the following entries:

```
[09/18/23]seed@VM:~/lab2$ getfacl manager_task1.txt
# file: manager_task1.txt
# owner: user3
# group: Manager
user::rwx
group::rwx
group:Manager:rwx
group:Board:rwx
mask::rwx
other::---
```

All board_plan*.txt files should look like this:



A screenshot of a terminal window titled "Terminal". The command entered is "getfacl board_plan1.txt". The output shows the file's access control list (ACL) with the following entries:

```
[09/18/23]seed@VM:~/lab2$ getfacl board_plan1.txt
# file: board_plan1.txt
# owner: user5
# group: Board
user::rwx
group::rwx
group:Board:rwx
mask::rwx
other::---
```

With permissions properly set in this scenario, we can move to checking/testing if they work as intended. The goals are as follows:

- user1 and user2 have full access rights to the employee_info1.txt file, and no access rights to the other group files.
- user3 and user4 have full access rights to the employee_info1.txt file and manager_task1.txt file, no access rights to the Board group files.
- user5 and user6 have full access rights to files from all groups. Ex. employee_info1.txt, manager_task1.txt, and board_plan1.txt.

Check access for user1 (Employee, owner):

```
sudo -u user1 cat employee_info1.txt # Should work  
sudo -u user1 cat manager_task1.txt # Should not work  
sudo -u user1 cat board_plan1.txt # Should not work
```

Test:

Read passed:

```
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat employee_info1.txt  
This is a file for user1 in the group Employee, edit  
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat manager_task1.txt  
cat: manager_task1.txt: Permission denied  
[09/18/23]seed@VM:~/lab2$ sudo -u user1 cat board_plan1.txt  
cat: board_plan1.txt: Permission denied
```

Write passed (checked using vim):

```
[09/18/23]seed@VM:~/lab2$ sudo -u user1 vim employee_info1.txt  
[09/18/23]seed@VM:~/lab2$ sudo -u user1 vim manager_task1.txt  
[09/18/23]seed@VM:~/lab2$ sudo -u user1 vim board_plan1.txt
```

```
This is a file for user1 in the group Employee, edit  
~  
~  
~  
~  
"employee_info1.txt" 1L, 53C 1,1 All
```

```
~  
~  
~  
~  
"manager_task1.txt" [Permission Denied] 0,0-1 All
```

```
~  
~  
~  
~  
"board_plan1.txt" [Permission Denied] 0,0-1 All
```

Check access for user2 (Employee, not owner):

```
sudo -u user2 cat employee_info1.txt # Should work  
sudo -u user2 cat manager_task1.txt # Should not work  
sudo -u user2 cat board_plan1.txt # Should not work
```

Test:

Read passed:

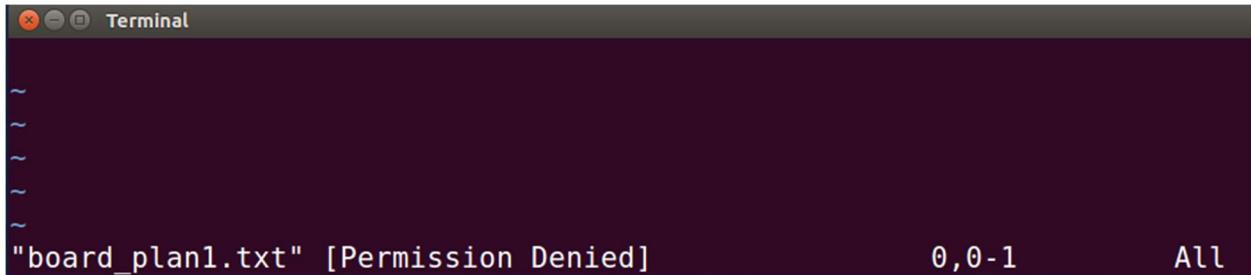
```
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat employee_info1.txt  
This is a file for user1 in the group Employee, edit  
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat manager_task1.txt  
cat: manager_task1.txt: Permission denied  
[09/18/23]seed@VM:~/lab2$ sudo -u user2 cat board_plan1.txt  
cat: board_plan1.txt: Permission denied
```

Write passed (checked using vim):

```
[09/18/23]seed@VM:~/lab2$ sudo -u user2 vim employee_info1.txt  
[09/18/23]seed@VM:~/lab2$ sudo -u user2 vim manager_task1.txt  
[09/18/23]seed@VM:~/lab2$ sudo -u user2 vim board_plan1.txt
```

```
[09/18/23]seed@VM:~/lab2$  
This is a file for user1 in the group Employee, edit  
~  
~  
~  
~  
~  
~  
"employee_info1.txt" 1L, 53C 1,1 All
```

```
[09/18/23]seed@VM:~/lab2$  
~  
~  
~  
~  
~  
~  
"manager_task1.txt" [Permission Denied] 0,0-1 All
```



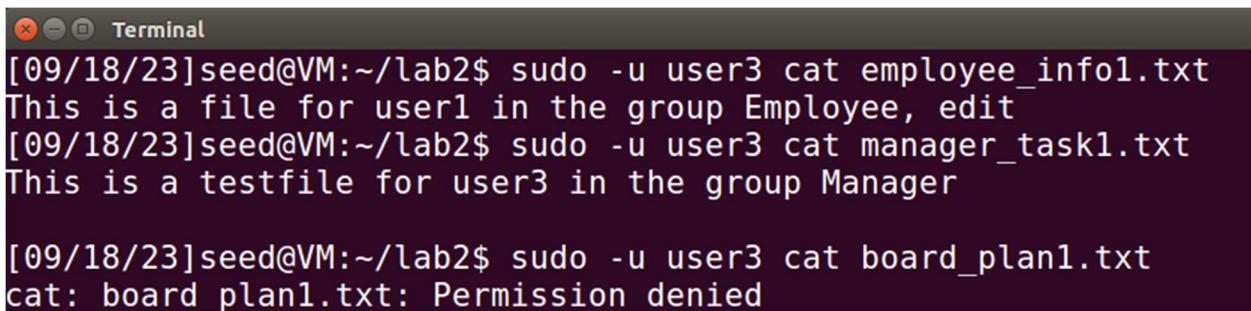
A screenshot of a Mac OS X terminal window titled "Terminal". The window shows a command-line interface with several tilde (~) symbols on the left. On the right, there is a status bar with the text "[Permission Denied]" and file statistics "0, 0-1 All".

Check access for user3 (Manager, owner):

```
sudo -u user3 cat employee_info1.txt # Should work  
sudo -u user3 cat manager_task1.txt # Should work  
sudo -u user3 cat board_plan1.txt # Should not work
```

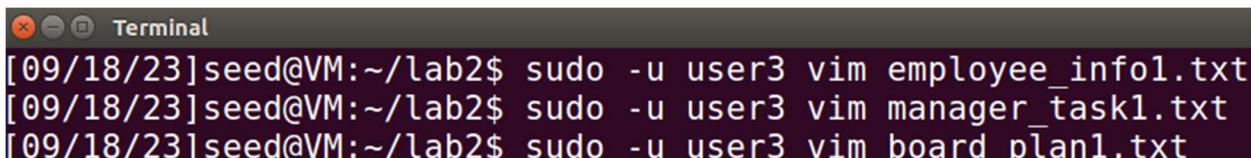
Test:

Read passed:

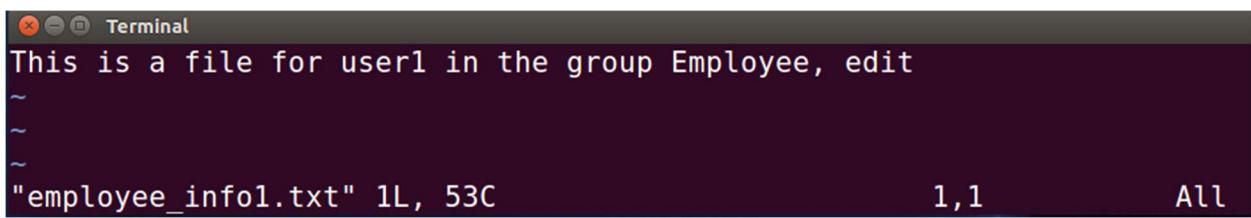


A screenshot of a Mac OS X terminal window titled "Terminal". The window shows three commands run by user3: reading employee_info1.txt, reading manager_task1.txt, and attempting to read board_plan1.txt. The first two succeed, while the third fails with a "cat: board_plan1.txt: Permission denied" error message.

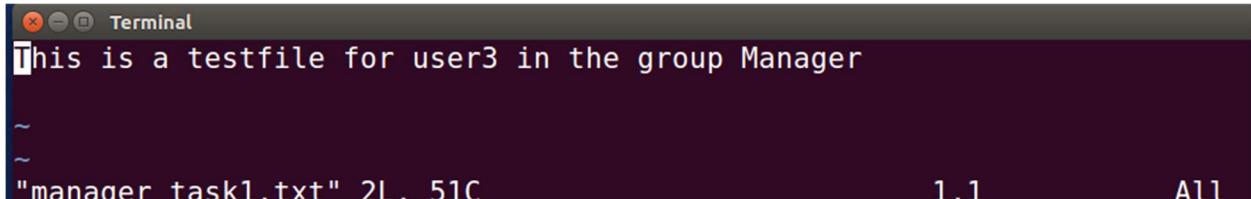
Write passed (checked using vim):



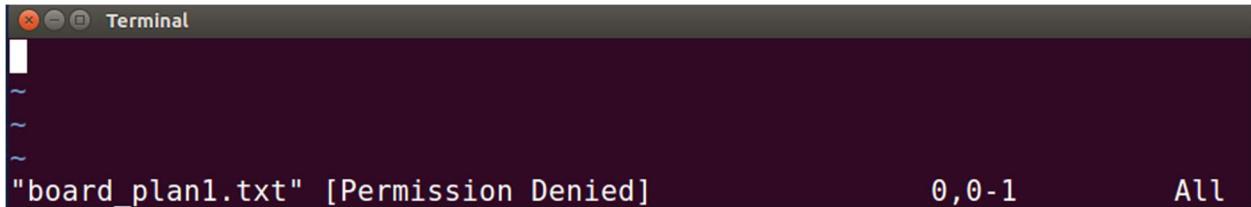
A screenshot of a Mac OS X terminal window titled "Terminal". The window shows three commands run by user3 using the vim editor on files employee_info1.txt, manager_task1.txt, and board_plan1.txt. All three commands complete successfully without errors.



A screenshot of a Mac OS X terminal window titled "Terminal". The window shows the contents of employee_info1.txt, which is a file for user1 in the group Employee, edit. The status bar indicates the file has 1L and 53C.



A screenshot of a Mac OS X terminal window titled "Terminal". The window shows the contents of manager_task1.txt, which is a testfile for user3 in the group Manager. The status bar indicates the file has 1L and 51C.



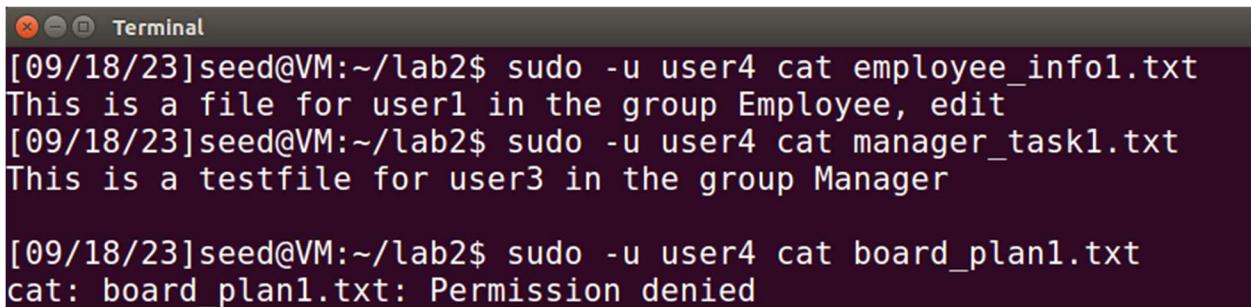
A screenshot of a terminal window titled "Terminal". The window shows a command being run: "board_plan1.txt" [Permission Denied]. The status bar at the bottom right indicates "0,0-1" and "All".

Check access for user4 (Manager, not owner):

```
sudo -u user4 cat employee_info1.txt # Should work  
sudo -u user4 cat manager_task1.txt # Should work  
sudo -u user4 cat board_plan1.txt # Should not work
```

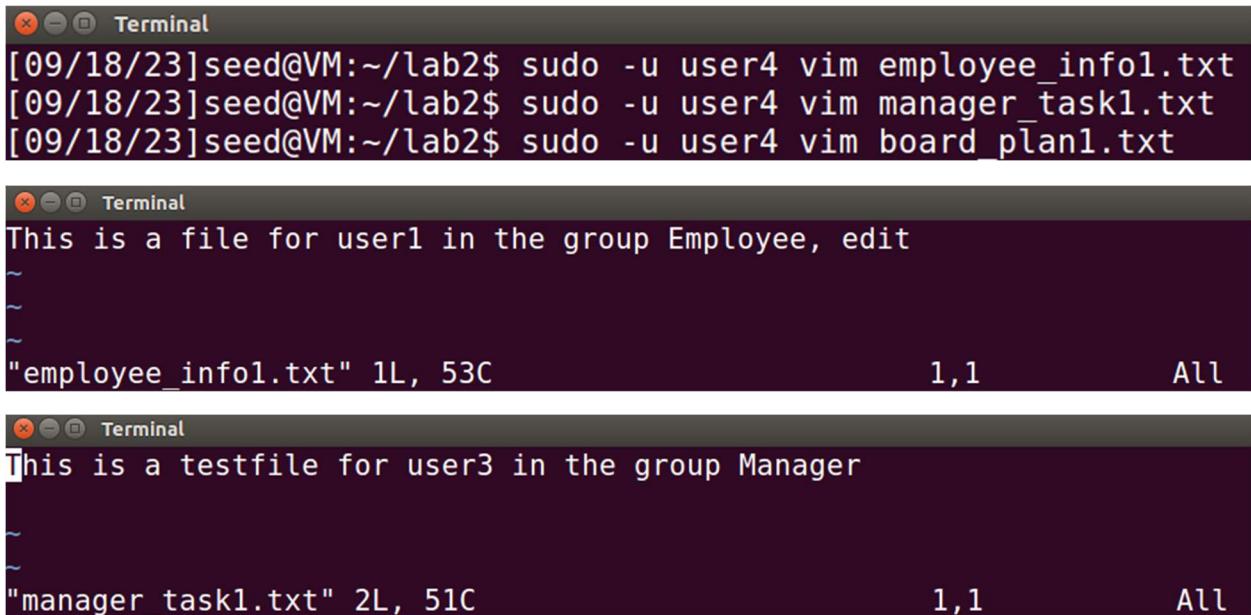
Test:

Read passed:

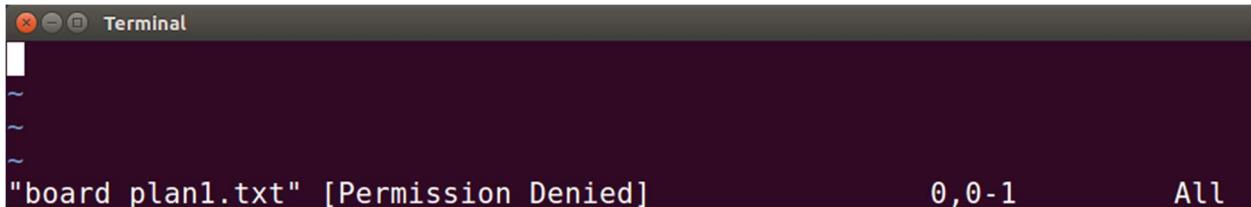


A screenshot of a terminal window titled "Terminal". It shows three commands run sequentially: "employee_info1.txt" (displaying its contents), "manager_task1.txt" (displaying its contents), and "board_plan1.txt" (which returns a "cat: board_plan1.txt: Permission denied" error).

Write passed (checked using vim):



Three screenshots of terminal windows titled "Terminal". The first window shows the creation and modification of "employee_info1.txt". The second window shows the creation and modification of "manager_task1.txt". The third window shows the creation and modification of "board_plan1.txt". All files were edited using the vim text editor.



A screenshot of a terminal window titled "Terminal". The window shows a command being run: "board plan1.txt". The output indicates a "Permission Denied" error. The status bar at the bottom right shows "0,0-1" and "All".

Check access for user5 (Board, owner):

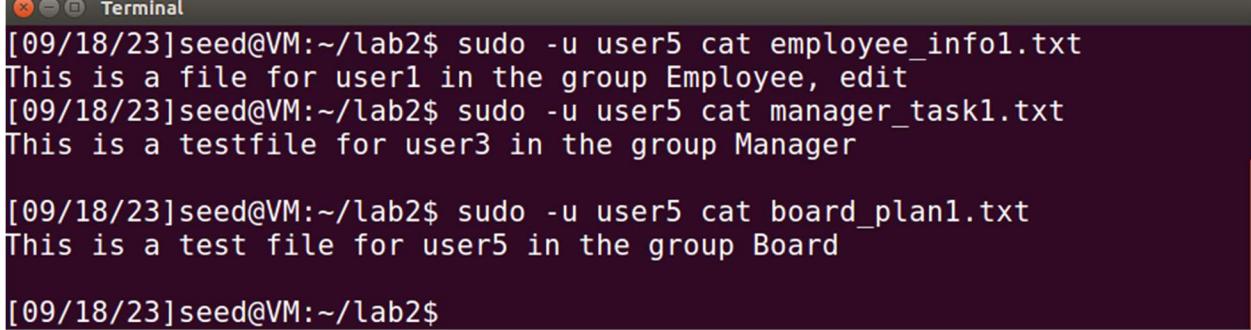
```
sudo -u user5 cat employee_info1.txt # Should work
```

```
sudo -u user5 cat manager_task1.txt # Should work
```

```
sudo -u user5 cat board_plan1.txt # Should work
```

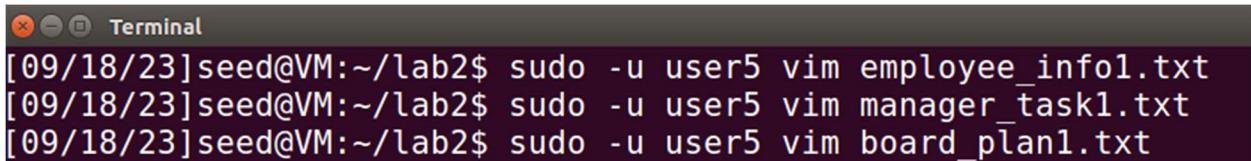
Test:

Read passed:

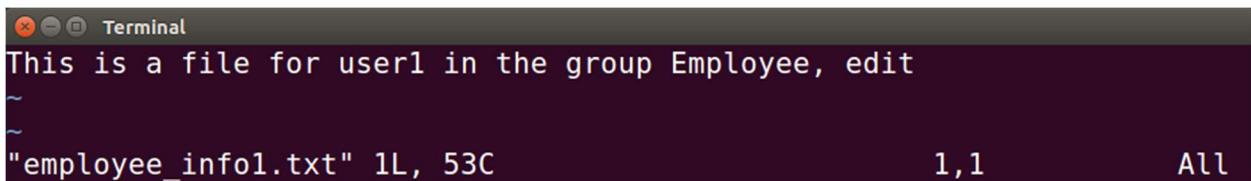


A screenshot of a terminal window titled "Terminal". It shows three commands being run: "sudo -u user5 cat employee_info1.txt", "sudo -u user5 cat manager_task1.txt", and "sudo -u user5 cat board_plan1.txt". Each command outputs a file content message. The status bar at the bottom right shows "0,0-1" and "All".

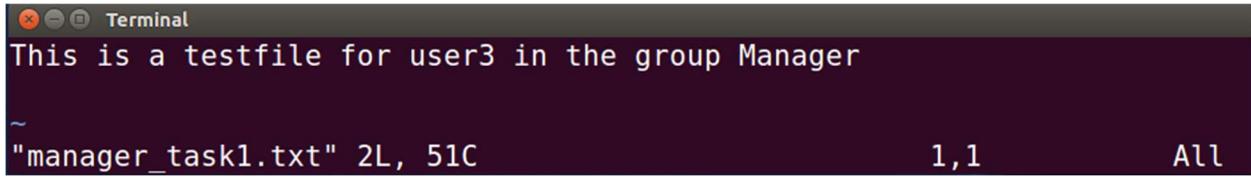
Write passed (checked using vim):



A screenshot of a terminal window titled "Terminal". It shows three commands being run: "sudo -u user5 vim employee_info1.txt", "sudo -u user5 vim manager_task1.txt", and "sudo -u user5 vim board_plan1.txt". The status bar at the bottom right shows "0,0-1" and "All".



A screenshot of a terminal window titled "Terminal". It displays the content of "employee_info1.txt": "This is a file for user1 in the group Employee, edit". The status bar at the bottom right shows "1,1" and "All".



A screenshot of a terminal window titled "Terminal". It displays the content of "manager_task1.txt": "This is a testfile for user3 in the group Manager". The status bar at the bottom right shows "1,1" and "All".

```
Terminal
This is a test file for user5 in the group Board
~
"board_plan1.txt" 2L, 50C                                     1,1          All
```

Check access for user6 (Board, not owner):

```
sudo -u user6 cat employee_info1.txt # Should work
sudo -u user6 cat manager_task1.txt # Should work
sudo -u user6 cat board_plan1.txt # Should work
```

Test:

Read passed:

```
Terminal
[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat employee_info1.txt
This is a file for user1 in the group Employee, edit
[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat manager_task1.txt
This is a testfile for user3 in the group Manager

[09/18/23]seed@VM:~/lab2$ sudo -u user6 cat board_plan1.txt
This is a test file for user5 in the group Board

[09/18/23]seed@VM:~/lab2$ █
```

Write passed (checked using vim):

```
Terminal
[09/18/23]seed@VM:~/lab2$ sudo -u user6 vim employee_info1.txt
[09/18/23]seed@VM:~/lab2$ sudo -u user6 vim manager_task1.txt
[09/18/23]seed@VM:~/lab2$ sudo -u user6 vim board_plan1.txt

Terminal
This is a file for user1 in the group Employee, edit
~
~
"employee_info1.txt" 1L, 53C                                     1,1          All

Terminal
This is a testfile for user3 in the group Manager

~
"manager_task1.txt" 2L, 51C                                     1,1          All

Terminal
This is a test file for user5 in the group Board

~
"board_plan1.txt" 2L, 50C                                     1,1          All
```

Results:

The multi-level security implementation outlined above was proven to correctly work and accomplished the goals that were established for testing. The hierarchy was successfully implemented where the Board group was able to have full rights for files belonging to the Board, Manager, and Employee groups. The Manager group was able to have full rights for files belonging to the Manager and Employee groups. Lastly, the Employee group was able to have full rights for files belonging to the Employee group. In my scenario, .txt files were used so execute was not applicable to test. However, based on the permissions established and the results of the test, it is reasonable to conclude that if execute was applicable that it would also function correctly in the permission hierarchy.