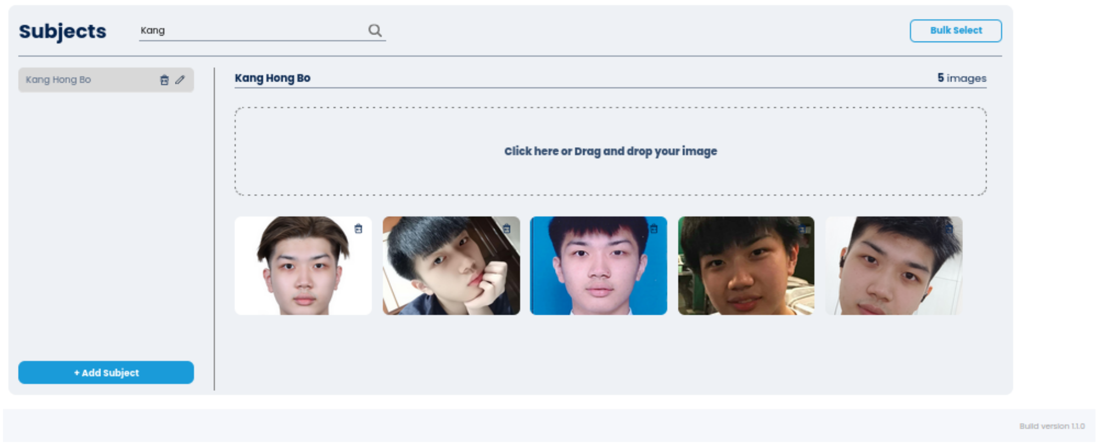
Assignment 2
Student ID: 32684673
Name: Kang Hong Bo

Task 1A

Face Collection



Testing Result
My photo

Photo	Similarity (with me)	Threshold = 0.95	Threshold = 0.98
1	1	True Positive	True Positive
2	1	True Positive	True Positive
3	0.99	True Positive	True Positive
4	1	True Positive	True Positive
5	1	True Positive	True Positive
6	1	True Positive	True Positive
7	1	True Positive	True Positive
8	0.99	True Positive	True Positive
9	1	True Positive	True Positive
10	1	True Positive	True Positive

The photo that isn't me

Photo	Similarity (with me)	Threshold = 0.95	Threshold = 0.98
1	N/A	True Negative	True Negative
2	N/A	True Negative	True Negative
3	N/A	True Negative	True Negative
4	N/A	True Negative	True Negative
5	0.99	False Positive	False Positive
6	N/A	True Negative	True Negative
7	N/A	True Negative	True Negative
8	N/A	True Negative	True Negative
9	N/A	True Negative	True Negative
10	0.98	False Positive	False Positive

Calculation of False Acceptance Rate and False Rejection Rate

Threshold: 0.95

False Positive = 2
True Positive = 10

True Negative = 8
False Negative = 0

FAR = FP /(FP+TP) = 2/(2+10)=0.167
FRR= FN / (FN+TP)=0/8=0

Threshold: 0.98

False Positive = 2
True Positive = 10

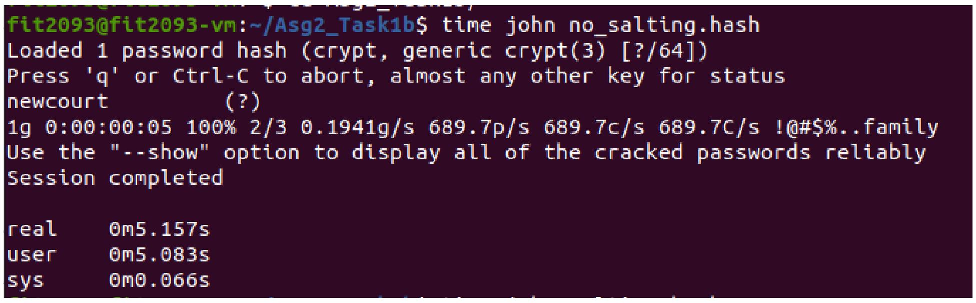
True Negative = 8
False Negative = 0

FAR = FP /(FP+TP) = 2/(2+10)=0.167
FRR= FN / (FN+TP)=0/8=0

The threshold determines the level of confidence required to accept or reject a claimed identity. This also means that it is a pointer that determines whether a user can access the system. The impact of the choice of the threshold is whether setting it higher or lower. If we let the threshold too low, it can increase the risk of attackers by allowing them to pass through the authentication process undetected. This would lead to data breaches, financial losses, and reputational damage. On the other hand, setting the threshold too high can result in false rejections which will cause a decrease in the usability of the system.

Task 1b

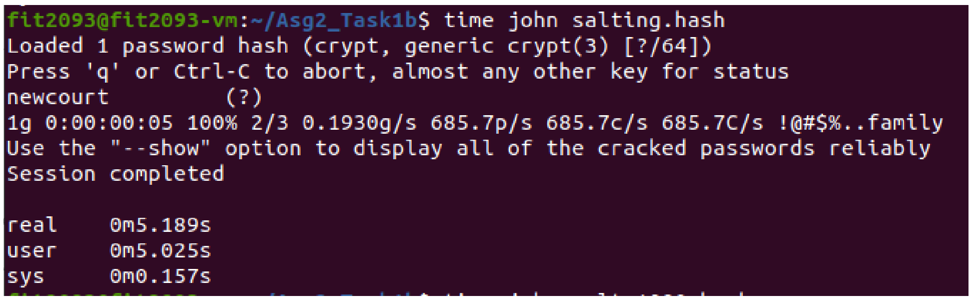
The no_slating.hash password



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time john no_salting.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt         (?)
1g 0:00:00:05 100% 2/3 0.1941g/s 689.7p/s 689.7c/s 689.7C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m5.157s
user    0m5.083s
sys     0m0.066s
```

User time took: 5.083s

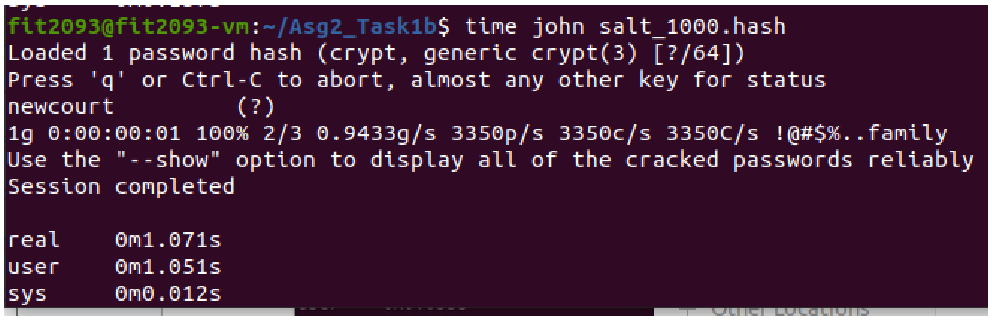
The salting.hash password



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time john salting.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt         (?)
1g 0:00:00:05 100% 2/3 0.1930g/s 685.7p/s 685.7c/s 685.7C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m5.189s
user    0m5.025s
sys     0m0.157s
```

System time took: 5.025s

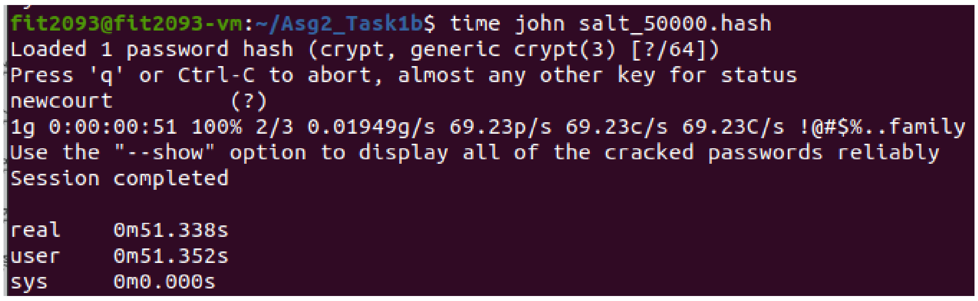
The salt_1000.hash password



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time john salt_1000.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt         (?)
1g 0:00:00:01 100% 2/3 0.9433g/s 3350p/s 3350c/s 3350C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m1.071s
user    0m1.051s
sys     0m0.012s
```

System time took: 1.051s

The salt_50000.hash password

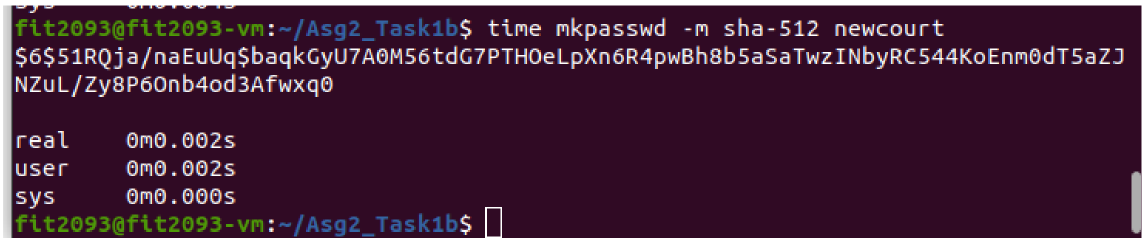


```
fit2093@fit2093-vm:~/Asg2_Task1b$ time john salt_50000.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
newcourt         (?)
1g 0:00:00:51 100% 2/3 0.01949g/s 69.23p/s 69.23c/s 69.23C/s !@#$%..family
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m51.338s
user    0m51.352s
sys     0m0.000s
```

System time took : 51.352s

Create password

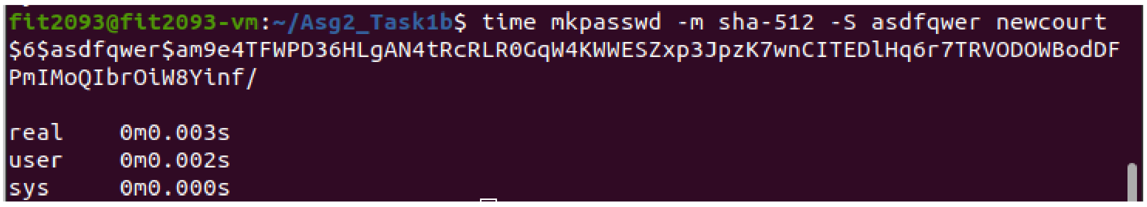
Password with no salt with default round



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 newcourt
$6$51RQja/naEuUq$baqkGyU7A0M56tdG7PTHOeLpXn6R4pwBh8b5aSaTwzINbyRC544KoEnm0dT5aZJ
NZuL/Zy8P6Onb4od3Afwxq0

real    0m0.002s
user    0m0.002s
sys     0m0.000s
fit2093@fit2093-vm:~/Asg2_Task1b$
```

User time : 0.002s

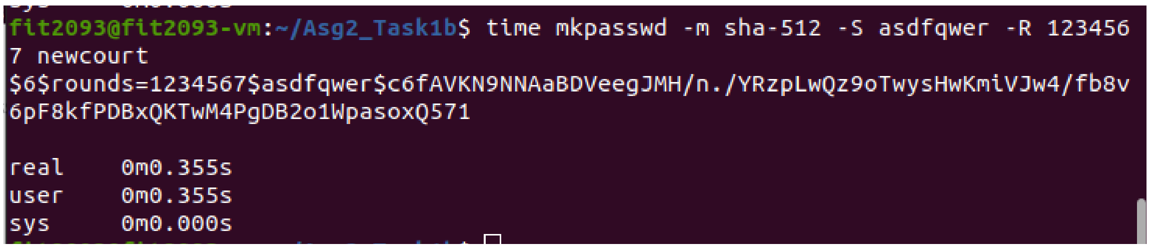
Password with salt and default round



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -S asdfqwer newcourt
$6$asdfqwer$am9e4TFWPD36HLgAN4tRcRLR0GqW4KWWESZxp3JpzK7wnCITEDlHq6r7TRVODOWBodDF
PmIMoQIbrOiW8Yinf/

real    0m0.003s
user    0m0.002s
sys     0m0.000s
```

User time : 0.002s

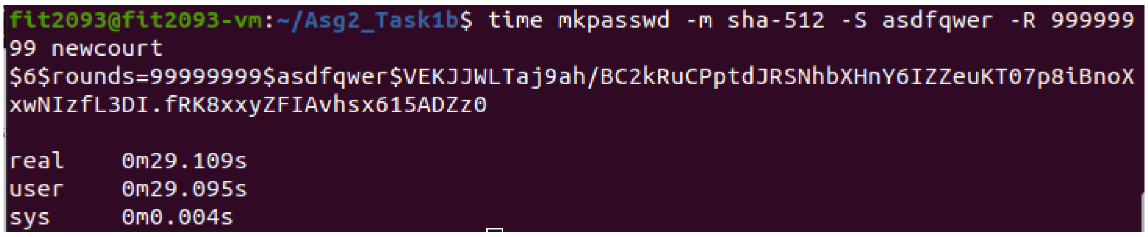
Password with salt and preferred round (1234567)



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -S asdfqwer -R 123456
7 newcourt
$6$rounds=1234567$asdfqwer$c6fAVKN9NNAaBDVeegJMH/n./YRzpLwQz9oTwysHwKmiVJw4/fb8v
6pF8kfPDBxQKTwM4PgDB2o1WpasoxQ571

real    0m0.355s
user    0m0.355s
sys     0m0.000s
```

User time : 0.355s

Password with salt and higher round (99999999)



```
fit2093@fit2093-vm:~/Asg2_Task1b$ time mkpasswd -m sha-512 -S asdfqwer -R 999999
99 newcourt
$6$rounds=99999999$asdfqwer$VEKJJWLTaj9ah/BC2kRuCPptdJRSNhbXHnY6IZZeuKT07p8iBnoX
xwNIzfL3DI.fRK8xxyZFIAvhsx615ADZz0

real    0m29.109s
user    0m29.095s
sys     0m0.004s
```

User time: 29.095s

So by comparing the user time recorded above we can see John the Ripper function is directly affected by the round of hashing of the password. The password with only 1000 rounds is being brute forced with the shortest time since the no salt password and salted password use default round which is 5000 is higher than the salt function of 1000 rounds beside the password of 50000 rounds takes the most time to brute force the password. But making a password with salt and round doesn't take much time, even if we have many rounds like the example above which only takes 29.095s. The round is way larger than the password with a round of 50000 which needs 51.352s to brute force.

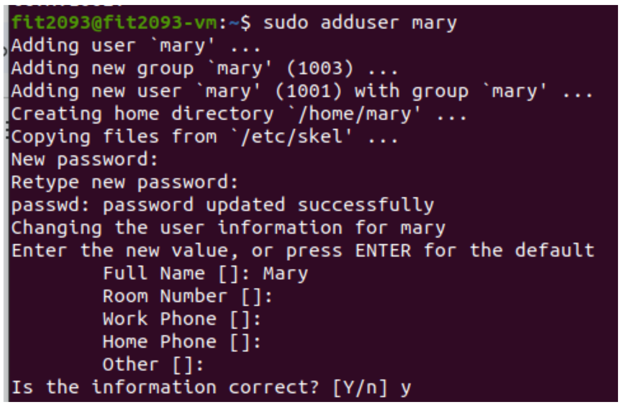
Estimating the time taken for a brute-force search through a dictionary of 200 million

1. Password with no salt with default round
   5.083s * 200million = 1016600000s=3.22361745307 decades
2. Password with salt with default round
   5.025s * 200million = 1005000000s=3.186834094368 decades
3. Password with salt with 1000 rounds
   1.051s * 200million = 210200000s=6.66539827499 years
4. Password with salt with 50000 rounds
   51.352s*200million= 10270400000s=3.2567224759006 centuries

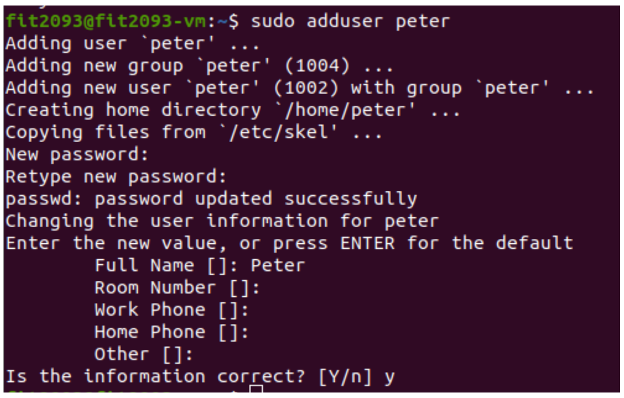
So for my recommendation we should use the salted hash function with a large round which will make brute forcing much harder since having salt and large rounds doesn't cost much time and it provides more security compared to the salt-only hashed password.

Task 2

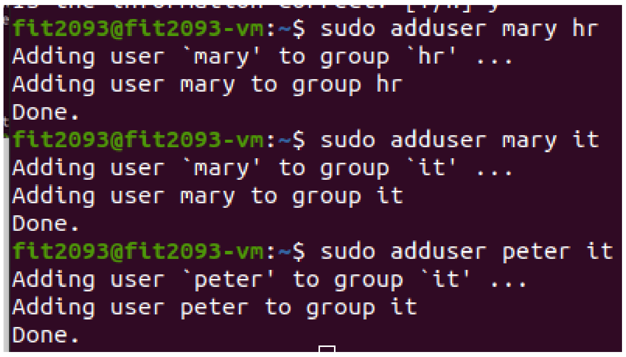
Add users and add them to the group.



```
fit2093@fit2093-vm:~$ sudo adduser mary
Adding user `mary' ...
Adding new group `mary' (1003) ...
Adding new user `mary' (1001) with group `mary' ...
Creating home directory `/home/mary' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for mary
Enter the new value, or press ENTER for the default
        Full Name []: Mary
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

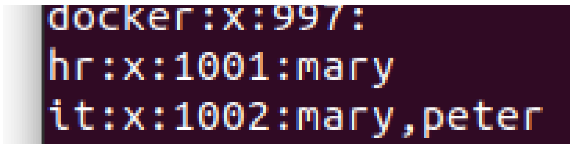


```
fit2093@fit2093-vm:~$ sudo adduser peter
Adding user `peter' ...
Adding new group `peter' (1004) ...
Adding new user `peter' (1002) with group `peter' ...
Creating home directory `/home/peter' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for peter
Enter the new value, or press ENTER for the default
        Full Name []: Peter
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```



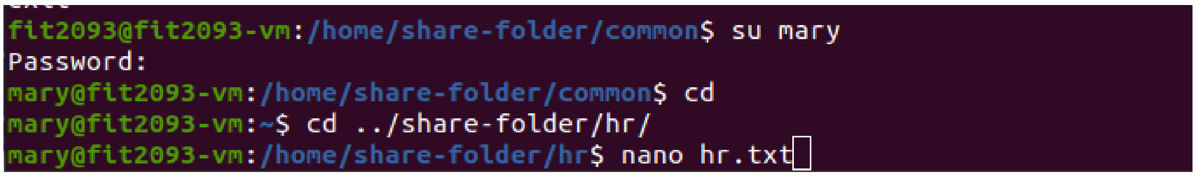
```
fit2093@fit2093-vm:~$ sudo adduser mary hr
Adding user `mary' to group `hr' ...
Adding user mary to group hr
Done.
fit2093@fit2093-vm:~$ sudo adduser mary it
Adding user `mary' to group `it' ...
Adding user mary to group it
Done.
fit2093@fit2093-vm:~$ sudo adduser peter it
Adding user `peter' to group `it' ...
Adding user peter to group it
Done.
```

By using cat /etc/group we can see that they r in the group



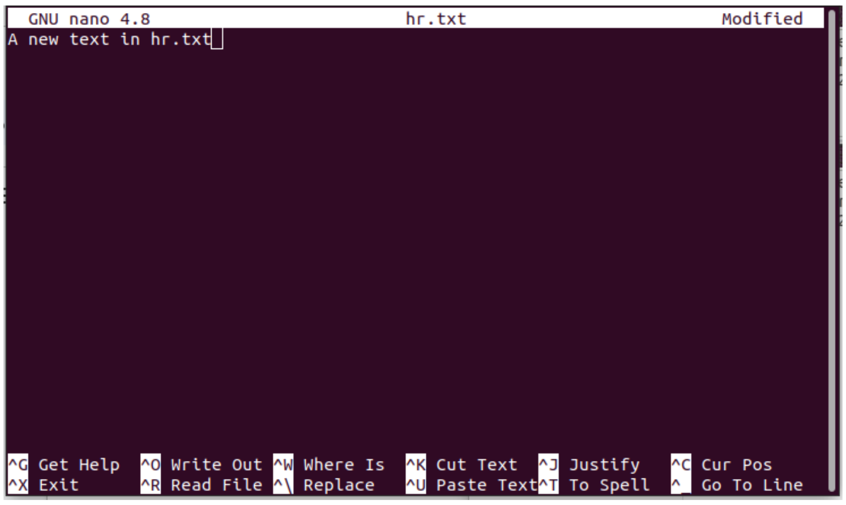
```
docker:x:997:
hr:x:1001:mary
it:x:1002:mary,peter
```

Now we switch users to mary by using mary. With (su mary)



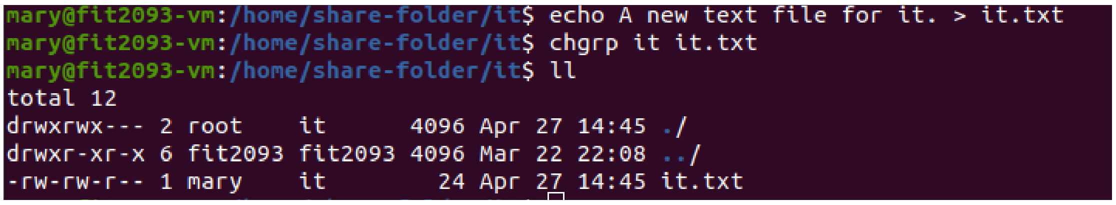
```
fit2093@fit2093-vm:/home/share-folder/common$ su mary
Password:
mary@fit2093-vm:/home/share-folder/common$ cd
mary@fit2093-vm:~$ cd ../share-folder/hr/
mary@fit2093-vm:/home/share-folder/hr$ nano hr.txt
```

A modified text is written by using nano hr.txt



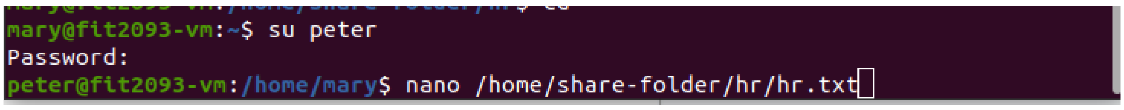
```
  GNU nano 4.8                    hr.txt                        Modified
A new text in hr.txt

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

After this we need to cd to it directory which can just (cd ../it)
Echo for a new text file to it and change the ownership of the text file to the group.

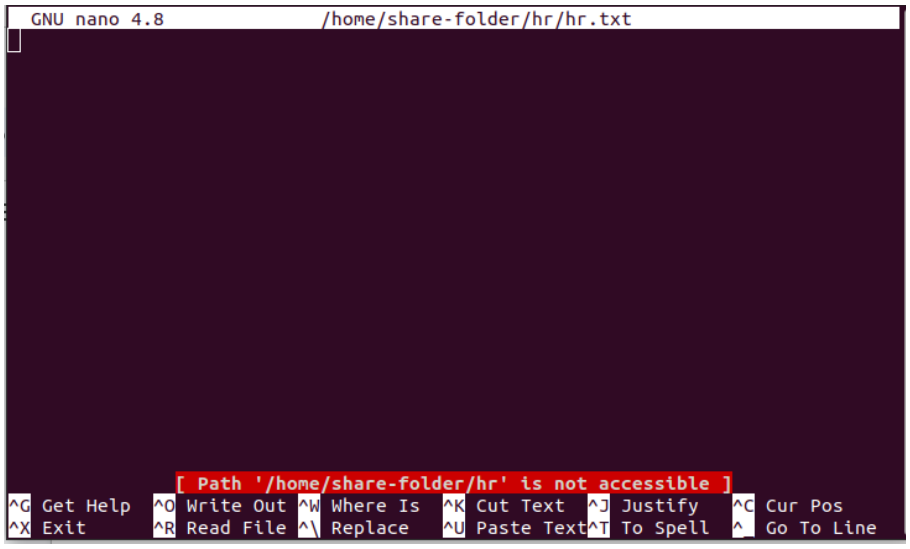


```
mary@fit2093-vm:/home/share-folder/it$ echo A new text file for it. > it.txt
mary@fit2093-vm:/home/share-folder/it$ chgrp it it.txt
mary@fit2093-vm:/home/share-folder/it$ ll
total 12
drwxrwx--- 2 root    it      4096 Apr 27 14:45 ./
drwxr-xr-x 6 fit2093 fit2093 4096 Mar 22 22:08 ../
-rw-rw-r-- 1 mary    it        24 Apr 27 14:45 it.txt
```

Peter phase:
Switch user to peter and modify the hr.txt



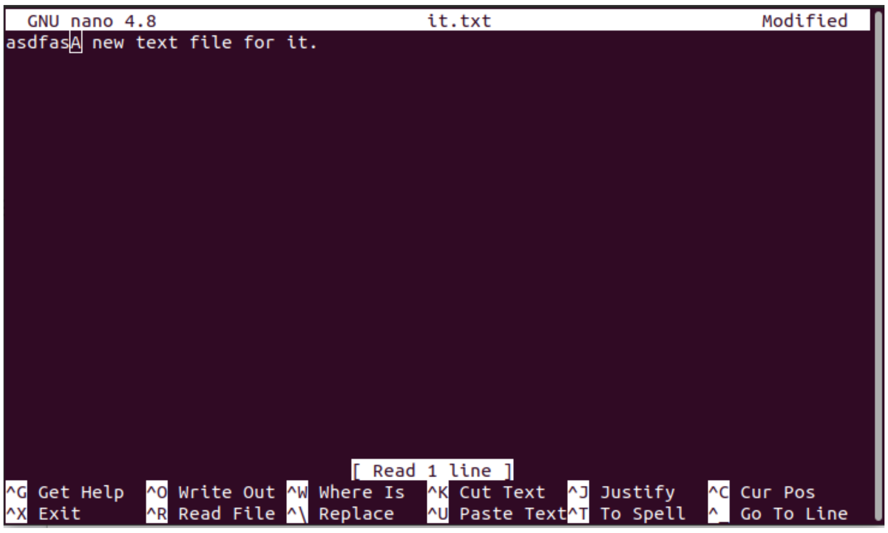
```
mary@fit2093-vm:/home/share-folder/hr$ cd
mary@fit2093-vm:~$ su peter
Password:
peter@fit2093-vm:/home/mary$ nano /home/share-folder/hr/hr.txt
```



```
  GNU nano 4.8              /home/share-folder/hr/hr.txt


             [ Path '/home/share-folder/hr' is not accessible ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

We can see that the directory to hr is not accessible since peter is not in a group of hr.

Then try to modify the it.txt

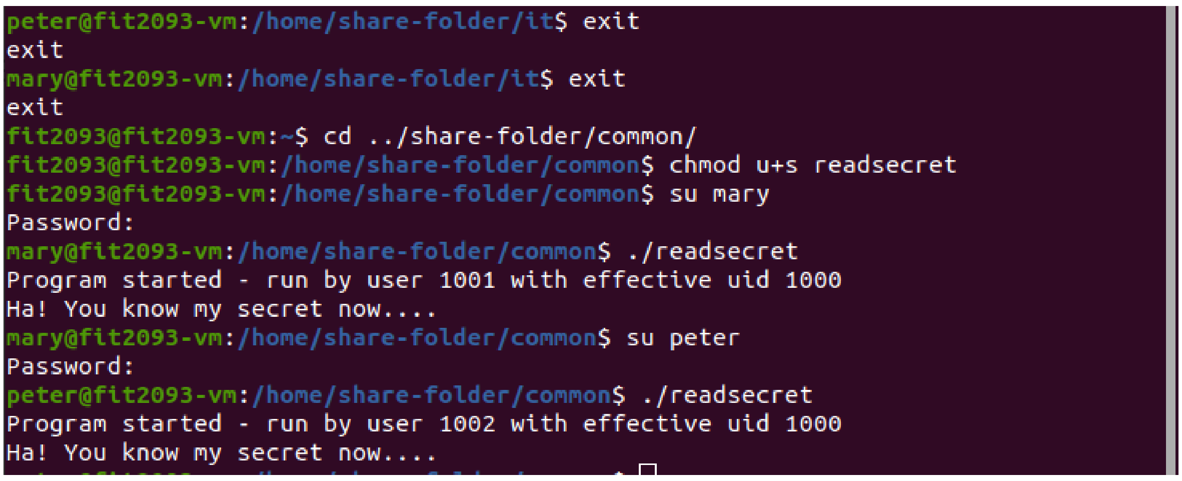


```
  GNU nano 4.8                        it.txt                        Modified
asdfasA new text file for it.

                                    [ Read 1 line ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

We can read and modify the text file as we are in the group of it

Task 2b

Now we switch back to the fit2093 and set UID to the program by using chmod u+s readsecret.

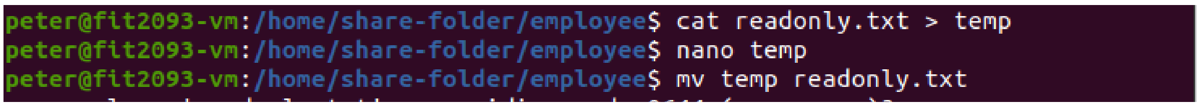


```
peter@fit2093-vm:/home/share-folder/it$ exit
exit
mary@fit2093-vm:/home/share-folder/it$ exit
exit
fit2093@fit2093-vm:~$ cd ../share-folder/common/
fit2093@fit2093-vm:/home/share-folder/common$ chmod u+s readsecret
fit2093@fit2093-vm:/home/share-folder/common$ su mary
Password:
mary@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1001 with effective uid 1000
Ha! You know my secret now....
mary@fit2093-vm:/home/share-folder/common$ su peter
Password:
peter@fit2093-vm:/home/share-folder/common$ ./readsecret
Program started - run by user 1002 with effective uid 1000
Ha! You know my secret now....
```

By using chmod u+s on an executable file other users can execute the file with the privileges of the file owner.

Task 2c

Since we have the write access to the directory by checking on user Peter
We can read the text and copy to a buffer then change the text in the buffer and replace the buffer to the readonly.txt file



```
peter@fit2093-vm:/home/share-folder/employee$ cat readonly.txt > temp
peter@fit2093-vm:/home/share-folder/employee$ nano temp
peter@fit2093-vm:/home/share-folder/employee$ mv temp readonly.txt
```

To mitigate this problem we can also let the directory read-only for the other user and only hr have the write permission which will solve this problem.