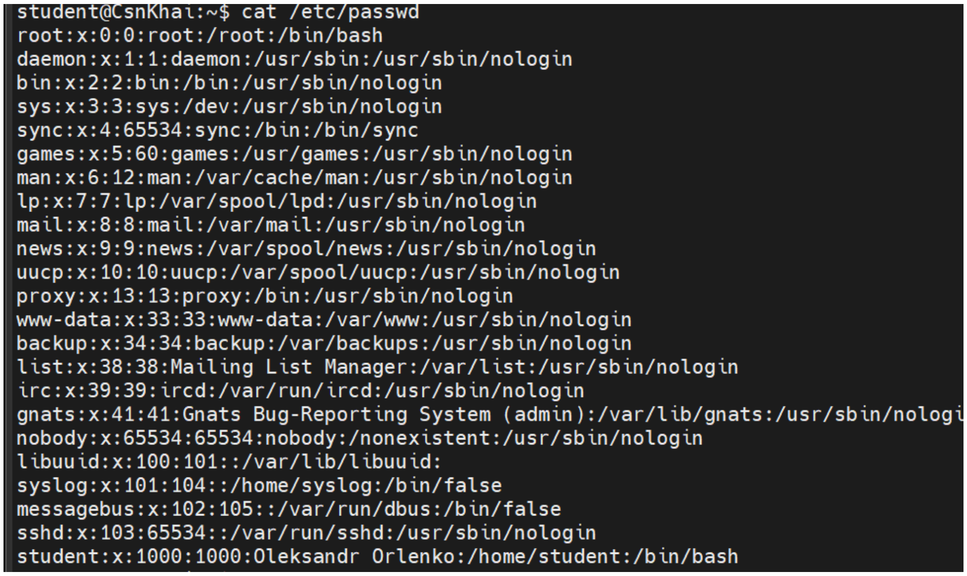
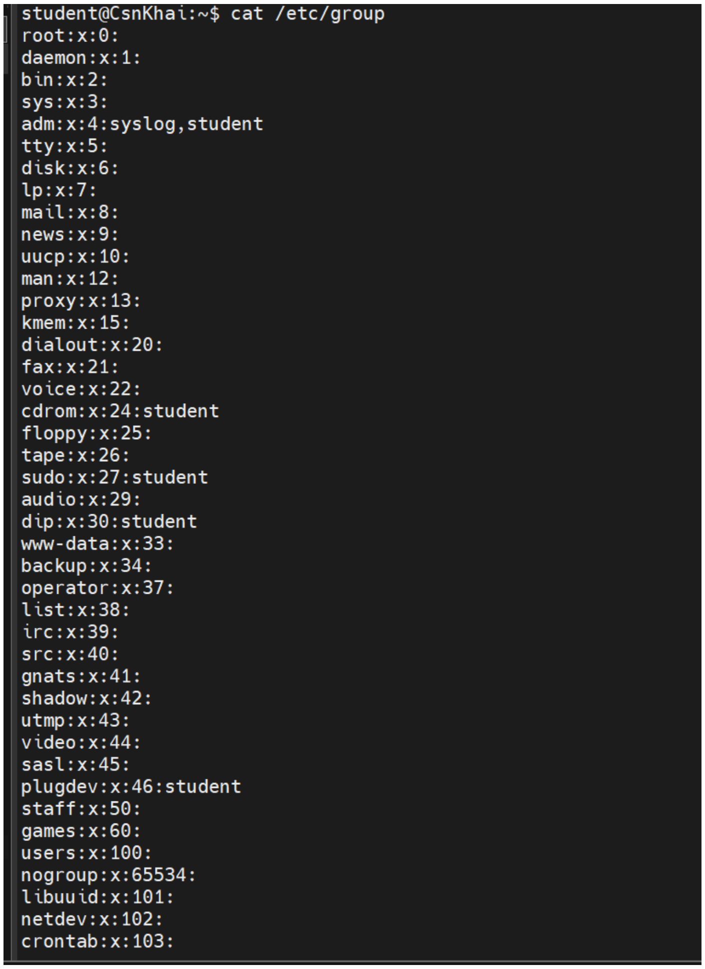
Task assignment.

Task 1



```
student@CsnKhai:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:105::/var/run/dbus:/bin/false
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
student:x:1000:1000:Oleksandr Orlenko:/home/student:/bin/bash
```



```
student@CsnKhai:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,student
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:student
floppy:x:25:
tape:x:26:
sudo:x:27:student
audio:x:29:
dip:x:30:student
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:
sasl:x:45:
plugdev:x:46:student
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
libuuid:x:101:
netdev:x:102:
crontab:x:103:
```

Task 2

UID stands for User Identifier. It is a unique numeric identifier assigned to each user account in a Unix-like operating system, including Linux. UID is used to distinguish between different users in the system.

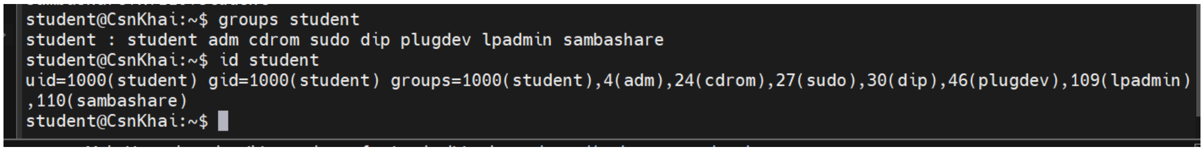
- System Users: 0 to 99
- Regular Users: 100 to 65535 (or lower depending on the system)
- Reserved and Special Users: Typically, UIDs above 65535

Task 3

GID stands for Group Identifier. It is a unique numeric identifier assigned to each group in a Unix-like operating system, such as Linux. GIDs are used to categorize users into different groups, allowing the assignment of specific permissions and access rights to files and resources.

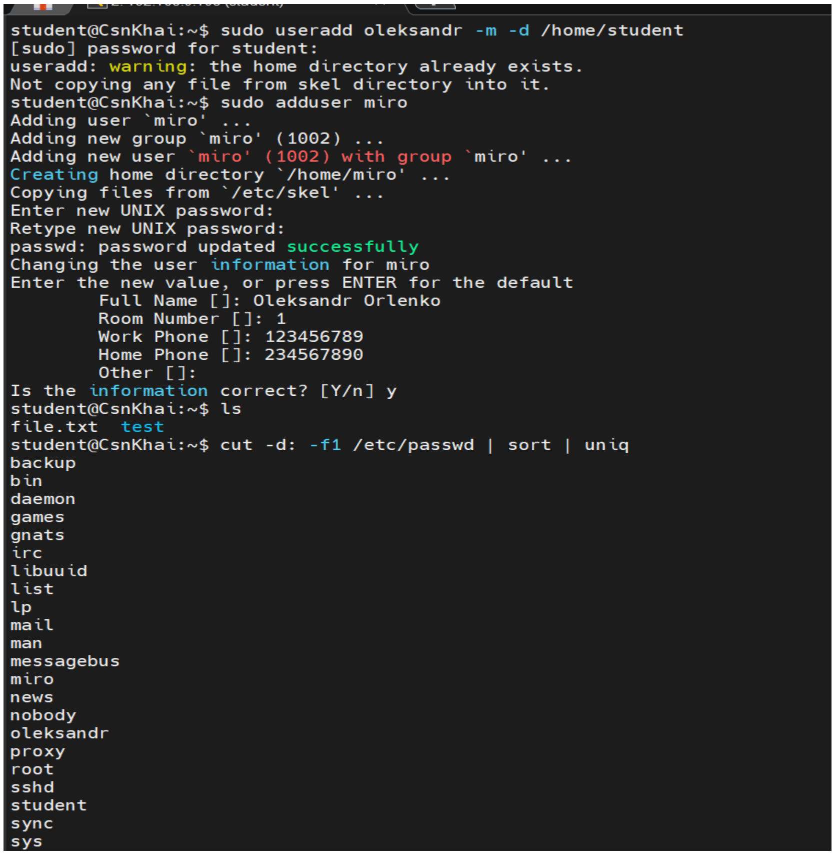
Defining a GID is similar to defining a UID when creating a new group or user. You can specify a GID using the **groupadd** command with the **-g** option.

Task 4

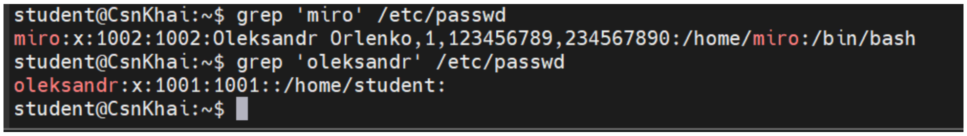


```
student@CsnKhai:~$ groups student
student : student adm cdrom sudo dip plugdev lpadmin sambashare
student@CsnKhai:~$ id student
uid=1000(student) gid=1000(student) groups=1000(student),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin)
,110(sambashare)
student@CsnKhai:~$ █
```

Task 5

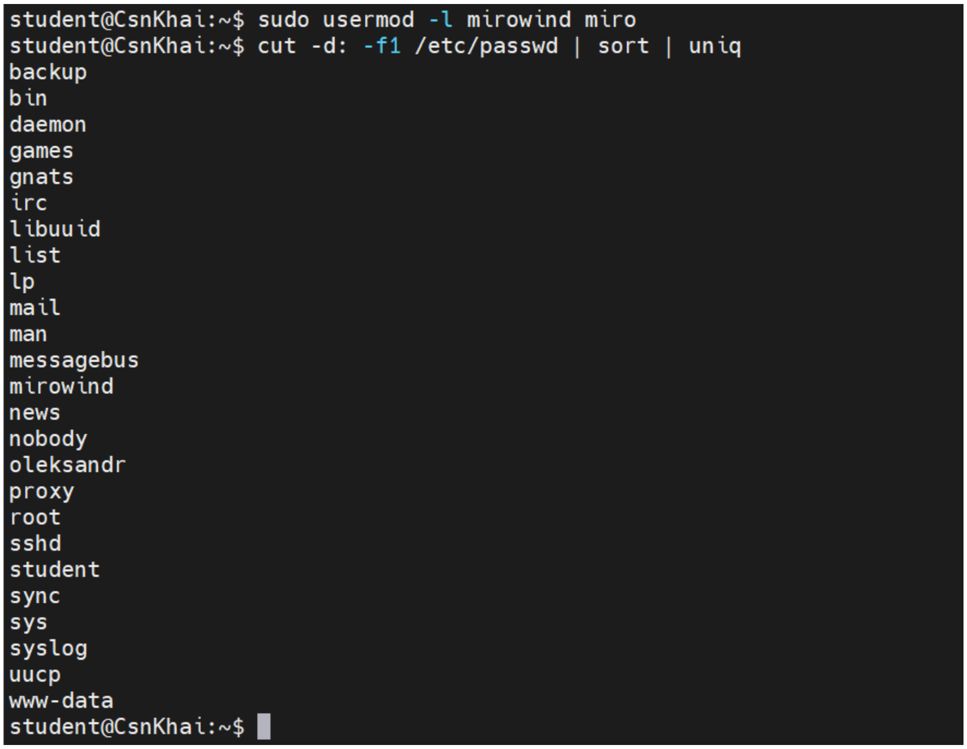


```
student@CsnKhai:~$ sudo useradd oleksandr -m -d /home/student
[sudo] password for student:
useradd: warning: the home directory already exists.
Not copying any file from skel directory into it.
student@CsnKhai:~$ sudo adduser miro
Adding user `miro' ...
Adding new group `miro' (1002) ...
Adding new user `miro' (1002) with group `miro' ...
Creating home directory `/home/miro' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for miro
Enter the new value, or press ENTER for the default
        Full Name []: Oleksandr Orlenko
        Room Number []: 1
        Work Phone []: 123456789
        Home Phone []: 234567890
        Other []:
Is the information correct? [Y/n] y
student@CsnKhai:~$ ls
file.txt  test
student@CsnKhai:~$ cut -d: -f1 /etc/passwd | sort | uniq
backup
bin
daemon
games
gnats
irc
libuuid
list
lp
mail
man
messagebus
miro
news
nobody
oleksandr
proxy
root
sshd
student
sync
sys
```



```
student@CsnKhai:~$ grep 'miro' /etc/passwd
miro:x:1002:1002:Oleksandr Orlenko,1,123456789,234567890:/home/miro:/bin/bash
student@CsnKhai:~$ grep 'oleksandr' /etc/passwd
oleksandr:x:1001:1001::/home/student:
student@CsnKhai:~$ 
```

Task 6



```
student@CsnKhai:~$ sudo usermod -l mirowind miro
student@CsnKhai:~$ cut -d: -f1 /etc/passwd | sort | uniq
backup
bin
daemon
games
gnats
irc
libuuid
list
lp
mail
man
messagebus
mirowind
news
nobody
oleksandr
proxy
root
sshd
student
sync
sys
syslog
uucp
www-data
student@CsnKhai:~$ 
```

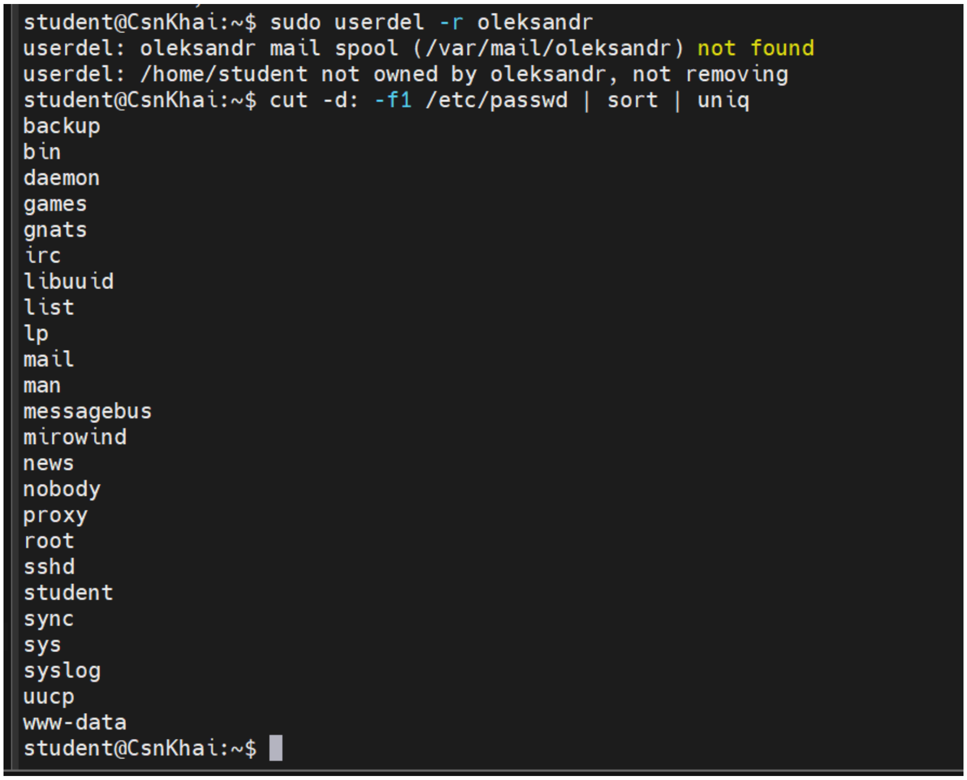
Task 7

skel_dir (or **skel** directory) stands for "skeleton directory" and is used in Linux to set up an initial set of files and configurations for newly created users. When you create a new user, the system copies the contents of the **skel_dir** to the home directory of the new user.

Typically, the **skel_dir** is located in the **/etc/skel** directory on most Linux systems. It can contain various files and folders such as:

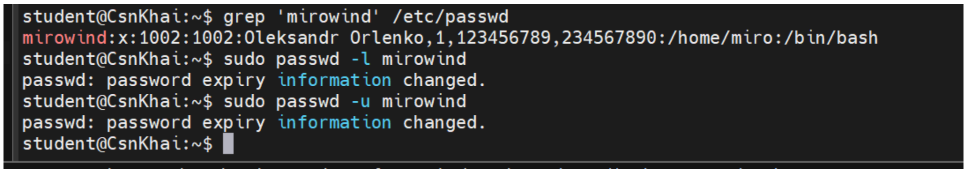
- **.bashrc** or **.bash_profile**: Files for configuring the Bash shell for the new user.
- **.profile**: A file executed when the user logs into the system.
- **.config**: A folder that may contain settings for applications and the working environment.
- Some additional files and folders that could be set up for the user's initial configuration.

Task 8

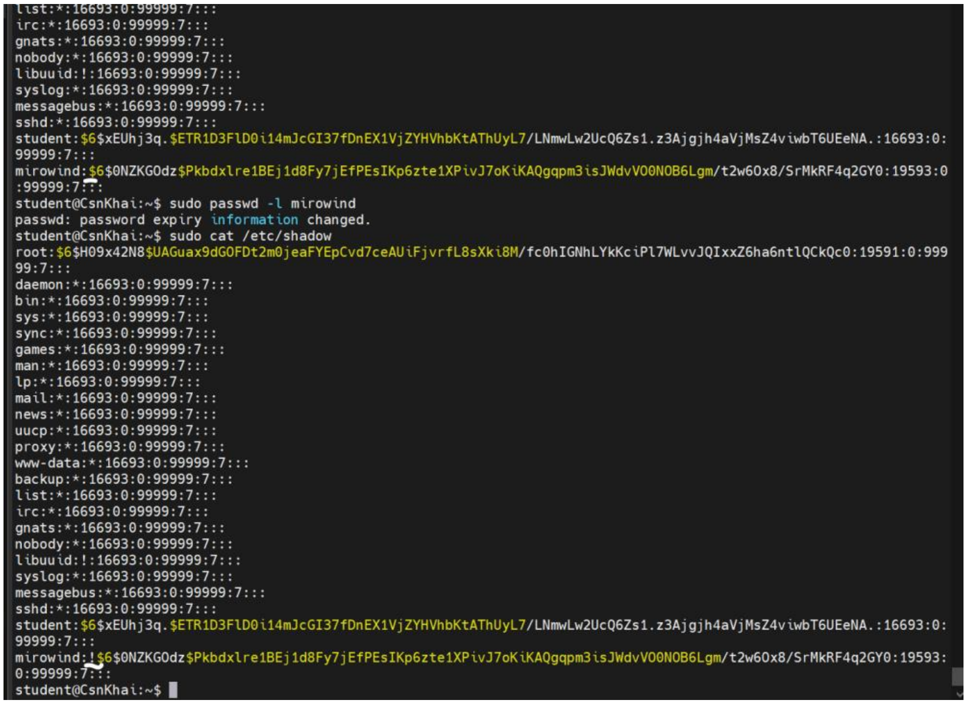


```
student@CsnKhai:~$ sudo userdel -r oleksandr
userdel: oleksandr mail spool (/var/mail/oleksandr) not found
userdel: /home/student not owned by oleksandr, not removing
student@CsnKhai:~$ cut -d: -f1 /etc/passwd | sort | uniq
backup
bin
daemon
games
gnats
irc
libuuid
list
lp
mail
man
messagebus
mirowind
news
nobody
proxy
root
sshd
student
sync
sys
syslog
uucp
www-data
student@CsnKhai:~$
```

Task 9

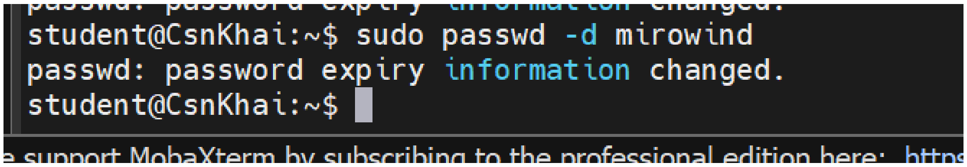


```
student@CsnKhai:~$ grep 'mirowind' /etc/passwd
mirowind:x:1002:1002:Oleksandr Orlenko,1,123456789,234567890:/home/miro:/bin/bash
student@CsnKhai:~$ sudo passwd -l mirowind
passwd: password expiry information changed.
student@CsnKhai:~$ sudo passwd -u mirowind
passwd: password expiry information changed.
student@CsnKhai:~$
```



```
list:*:16693:0:99999:7:::
irc:*:16693:0:99999:7:::
gnats:*:16693:0:99999:7:::
nobody:*:16693:0:99999:7:::
libuuid:!:16693:0:99999:7:::
syslog:*:16693:0:99999:7:::
messagebus:*:16693:0:99999:7:::
sshd:*:16693:0:99999:7:::
student:$6$xEUhj3q.$ETR1D3FlD0i14mJcGI37fDnEX1VjZYHVhbKtAThUyL7/LNmwLw2UcQ6Zs1.z3Ajgjh4aVjMsZ4viwbT6UEeNA.:16693:0:
99999:7:::
mirowind:$6$0NZKGOdz$Pkbdxlre1BEj1d8Fy7jEfPEsIKp6zte1XPivJ7oKiKAQgqpm3isJWdvVO0NOB6Lgm/t2w6Ox8/SrMkRF4q2GY0:19593:0
:99999:7:::
student@CsnKhai:~$ sudo passwd -l mirowind
passwd: password expiry information changed.
student@CsnKhai:~$ sudo cat /etc/shadow
root:$6$H09x42N8$UAGuax9dGOFDt2m0jeaFYEpCvd7ceAUiFjvrfL8sXki8M/fc0hIGNhLYkKciPl7WLvvJQIxxZ6ha6ntlQCkQc0:19591:0:999
99:7:::
daemon:*:16693:0:99999:7:::
bin:*:16693:0:99999:7:::
sys:*:16693:0:99999:7:::
sync:*:16693:0:99999:7:::
games:*:16693:0:99999:7:::
man:*:16693:0:99999:7:::
lp:*:16693:0:99999:7:::
mail:*:16693:0:99999:7:::
news:*:16693:0:99999:7:::
uucp:*:16693:0:99999:7:::
proxy:*:16693:0:99999:7:::
www-data:*:16693:0:99999:7:::
backup:*:16693:0:99999:7:::
list:*:16693:0:99999:7:::
irc:*:16693:0:99999:7:::
gnats:*:16693:0:99999:7:::
nobody:*:16693:0:99999:7:::
libuuid:!:16693:0:99999:7:::
syslog:*:16693:0:99999:7:::
messagebus:*:16693:0:99999:7:::
sshd:*:16693:0:99999:7:::
student:$6$xEUhj3q.$ETR1D3FlD0i14mJcGI37fDnEX1VjZYHVhbKtAThUyL7/LNmwLw2UcQ6Zs1.z3Ajgjh4aVjMsZ4viwbT6UEeNA.:16693:0:
99999:7:::
mirowind:!$6$0NZKGOdz$Pkbdxlre1BEj1d8Fy7jEfPEsIKp6zte1XPivJ7oKiKAQgqpm3isJWdvVO0NOB6Lgm/t2w6Ox8/SrMkRF4q2GY0:19593:
0:99999:7:::
student@CsnKhai:~$
```

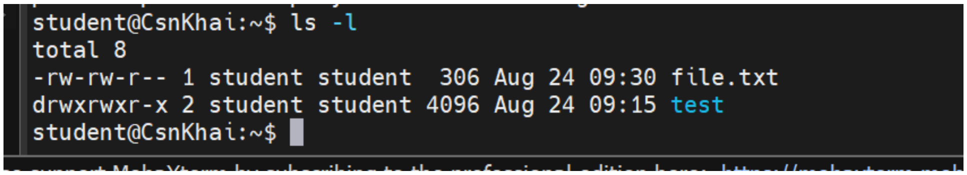
Task 10



```
passwd: password expiry information changed.
student@CsnKhai:~$ sudo passwd -d mirowind
passwd: password expiry information changed.
student@CsnKhai:~$
```

e support MobaXterm by subscribing to the professional edition here: https

Task 11



```
student@CsnKhai:~$ ls -l
total 8
-rw-rw-r-- 1 student student  306 Aug 24 09:30 file.txt
drwxrwxr-x 2 student student 4096 Aug 24 09:15 test
student@CsnKhai:~$
```

- **File Type and Permissions**: This column displays the file type and permissions. The first character indicates the file type (e.g., **-** for regular file, **d** for directory, **l** for symbolic link), followed by the file's permissions for owner, group, and others.
- **Number of Hard Links**: The number of hard links pointing to the file or directory.
- **Owner**: The owner of the file or directory.
- **Group**: The group associated with the file or directory.
- **File Size**: The size of the file in bytes.
- **Date and Time**: The date and time when the file was last modified.
- **File Name**: The name of the file or directory.

Task 12

Access rights in Linux are permissions that determine who can perform certain actions on a file or directory. The main roles or categories for which access rights are defined are:

- **Owner**: The user who owns the file or directory. They have the highest level of control and can modify permissions, delete the file, and read/write/execute it.
- **Group**: The group associated with the file or directory. All users in this group share the same permissions. Group members can also modify the file, depending on the permissions granted to the group.
- **Others**: All users who are not the owner and not in the group. This category includes all other users on the system. Their permissions are often the most restricted.

Access rights are represented using the following acronym:

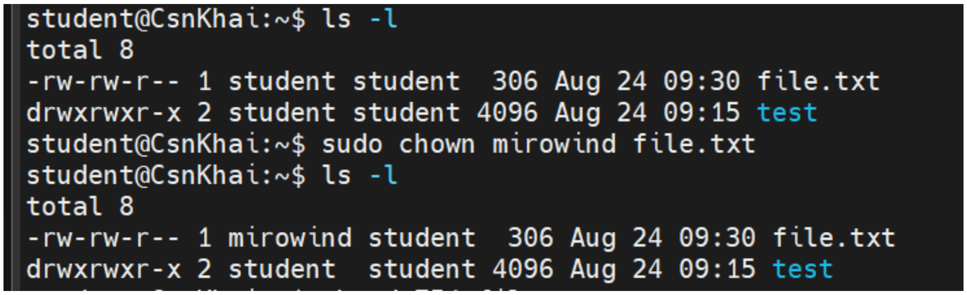
- **r**: Read permission. Allows viewing the contents of a file or listing the contents of a directory.
- **w**: Write permission. Allows modifying the contents of a file or creating/removing files in a directory.
- **x**: Execute permission. Allows running a file if it's a script or binary executable or accessing a directory's contents if it's executable.

Task 13

The sequence of defining the relationship between a file and a user involves understanding and assigning access permissions. This involves:

- **File Ownership**: Every file in Linux has an owner. The owner has the most control over the file and can modify its permissions. You can use the **chown** command to change the ownership of a file. For example, **sudo chown username filename**.
- **User Groups**: Files can also belong to user groups. Multiple users can be part of a group, and group permissions can be applied to files. You can use the **chgrp** command to change the group ownership of a file. For example, **sudo chgrp groupname filename**.
- **Access Permissions**: Once the ownership and group associations are set, you define the access permissions for the owner, group, and others using the **chmod** command. You can assign read (**r**), write (**w**), and execute (**x**) permissions for each category.
- **Effective Permissions**: The final relationship defines who can do what with the file. The effective permissions for a user are determined by the combination of their user category (owner, group, or others) and the permissions assigned to that category.

Task 14



```
student@CsnKhai:~$ ls -l
total 8
-rw-rw-r-- 1 student student  306 Aug 24 09:30 file.txt
drwxrwxr-x 2 student student 4096 Aug 24 09:15 test
student@CsnKhai:~$ sudo chown mirowind file.txt
student@CsnKhai:~$ ls -l
total 8
-rw-rw-r-- 1 mirowind student  306 Aug 24 09:30 file.txt
drwxrwxr-x 2 student  student 4096 Aug 24 09:15 test
```

Task 15

An octal representation of access rights is a way to express file permissions using three digits in base 8 (0-7), each representing a specific set of permissions for the owner, group, and others.

For example, in the octal representation **644**:

- The first digit (**6**) represents the owner's permissions, which are read (**4**), write (**2**), and no execute (**0**).
- The second digit (**4**) represents the group's permissions, which are read-only (**4**).
- The third digit (**4**) represents others' permissions, which are read-only (**4**).

The **umask** command in Linux is used to set the default permissions that will be applied to newly created files and directories. It works by subtracting the value you specify from the default maximum permission value (**777** for directories and **666** for files) to determine the actual permissions.

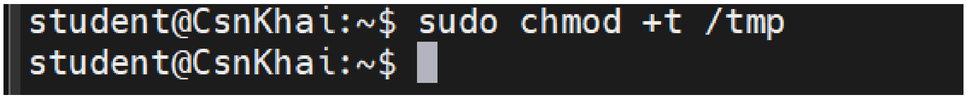
For instance, if you set a **umask** of **022**, the default permissions of new files would be **666 - 022 = 644**, and the default permissions of new directories would be **777 - 022 = 755**. The **umask** value is typically set in the shell configuration files like **.bashrc** or **.profile**.

Keep in mind that the **umask** value is subtracted from the maximum permission value, so a higher **umask** value will result in more restrictive permissions for newly created files and directories.

Task 16

Sticky Bit: The sticky bit is a special permission bit that can be set on a directory. It is typically used for directories that are shared among multiple users, such as /tmp. When the sticky bit is set on a directory, only the owner of a file can delete or move files from that directory, even if other users have write permission on that directory. This helps prevent accidental deletion or overwriting of files by other users.

ID Substitution Mechanism: The ID substitution mechanism allows performing actions on files and directories on behalf of another user or group if you have the appropriate permissions. This is useful for carrying out restricted tasks on behalf of users with limited privileges.



```
student@CsnKhai:~$ sudo chmod +t /tmp
student@CsnKhai:~$ 
```

Task 17

In a command script, there are several file attributes that are important for proper execution and security. These attributes include:

- **Executable Permission (x)**: The script file should have the executable permission set. This allows the script to be run as a program. You can use the **chmod** command to add the executable permission, for example: **chmod +x script.sh**.
- **Interpreter Shebang Line**: The script should begin with the interpreter shebang line that specifies the interpreter to be used for executing the script. For example, for a Bash script, the shebang line would be: **#!/bin/bash**. This tells the system which interpreter to use for interpreting the script's commands.
- **Correct Path to Commands**: If the script uses external commands or programs, their paths should be correctly specified. If the commands are not in the default system path, you should provide their full paths.

- **Readable Permission (r)**: The script file should have readable permission so that it can be opened and read by the interpreter. This is typically the default for most files.
- **Appropriate Ownership**: The script should be owned by the user who needs to execute it. Make sure the ownership of the script is appropriate to prevent unauthorized access.
- **Secure Permissions**: While the executable permission is necessary, be cautious about granting write permission to the script unless it's necessary, as this can potentially introduce security vulnerabilities.