User Management

User management in Linux involves creating, modifying, and deleting user accounts, as well as controlling their permissions and access to system resources.

A user in Linux is an entity that can create, modify, and delete resources, and perform other actions.

Each user has a unique User ID (UID)
Root User UID = 0
System Users UID = 1 to 999 (inclusive)
Local Users UID > 999

Question: Does this mean we can't create more than 1000 system users in Linux?
Answer: No, the system allows more than 1000 system users. The range UID 1 to 999 is reserved for system users, but if you try to add more, new system users will get a UID above 1000.

You can create up to 60,000 users in a single directory.

Root user is not used for everyday tasks. Normal users can gain root access through sudo.

By creating individual users, you can control who has access to the system and what they can do.

Each user has their own permissions for files, directories, and resources.

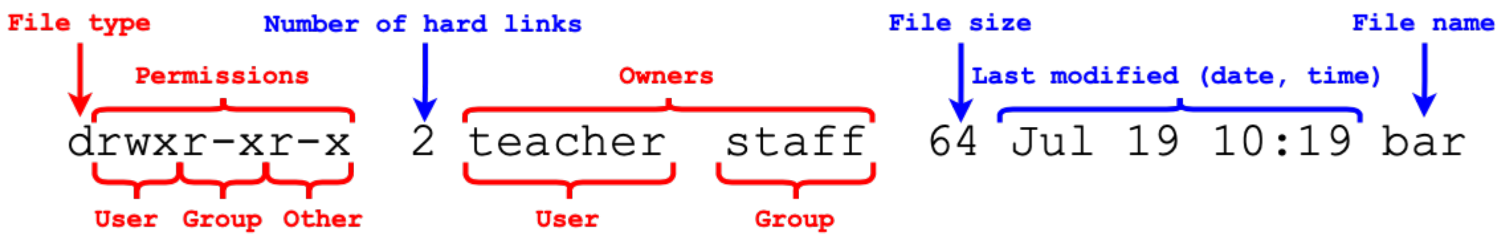
Users' actions are tracked, and if something goes wrong, you can check logs to see which user performed a certain action.

```
# To list out all the users in Linux
awk -F':' '{ print $1}' /etc/passwd

# id username
id username

# Add New User Low Level
sudo useradd username

# Add User alternative command - High Level
sudo adduser username

# Delete a User
sudo userdel -r username

# Assing Password to usernamer
passwd username

# Access User Configuartion
cat /etc/passwd

# Change Userid for the User
cat /etc/passwd

# Modify the Group Id of the User
usermod -g new_group_id username

# Change user login name
sudo usermod -l new_login_name old_login_name

# Change Home Directory
usermod -d new_home_directory_path username
```

File System

Linux file permissions control who can access files and directories. Here are the key commands to check permissions:

```
Command Description
ls -l foo.sh Check permissions of file foo.sh
ls -ld /var/log Check permissions of directory /var/log
```



```
Scope Symbol Description
User u The owner of the file or directory
Group g The group of users who can access the file
Other o Other users (world)
All a All users
```

Permission type	Symbol	If a file has this permission, you can:	If a directory has this permission, you can:
Read	r	Open and view file contents (e.g., `cat`, `head`, `tail`)	Read directory contents (e.g., `ls`, `du`)
Write	w	Edit, delete, or rename file (e.g., `nano`, `vi`)	Edit, delete, or rename directory/files within it; create files
Execute	x	Execute the file (e.g., `./deploy.sh`)	Enter the directory (e.g., `cd`); without `x`, `r` and `w` are useless
None	-	Do nothing	Do nothing

```
Command Description
chmod permission app.sh Change the permissions of a file app.sh
chown admin app.sh Change the owner of app.sh to admin
chgrp devs app.sh Change the group of app.sh to devs
umask Get a four-digit subtrahend
sudo Invoke superuser privileges
id Find your user ID (uid) and group ID
(gid)
groups Find all groups to which you belong
```

There are two methods to represent permissions on the command line: symbolic notation and octal notation. The first argument of the `chmod` command can use both representations.

This notation uses a combination of `[u/g/o/a]`, `[+/-/=]`, and `[r/w/x]` to change permissions.

Command in symbolic notation	Change in user (u) permissions	Change in group (g) permissions	Change in world (o) permissions
`chmod +x deploy.sh`	✓ Execute	✓ Execute	✓ Execute
`chmod a=x backup.sh`	☐ Read, ☐ Write, ✓ Execute	☐ Read, ☐ Write, ✓ Execute	☐ Read, ☐ Write, ✓ Execute
`chmod u-w config.yaml`	☐ Write	(No change)	(No change)
`chmod u+wx,g-x,o=rx install.sh`	✓ Write, ✓ Execute	☐ Execute	✓ Read, ☐ Write, ✓ Execute

Octal notation is a three-digit number ranging from 000 to 777, where each digit represents permissions as the sum of 4 (read), 2 (write), and 1 (execute).

Octal digit	Permission(s) granted	Symbolic
0	None	[u/g/o]-rwx
1	Execute permission only	[u/g/o]=x
2	Write permission only	[u/g/o]=w
3	Write and execute permissions only: 2 + 1 = 3	[u/g/o]=wx
4	Read permission only	[u/g/o]=r
5	Read and execute permissions only: 4 + 1 = 5	[u/g/o]=rx
6	Read and write permissions only: 4 + 2 = 6	[u/g/o]=rw
7	All permissions: 4 + 2 + 1 = 7	[u/g/o]=rwx

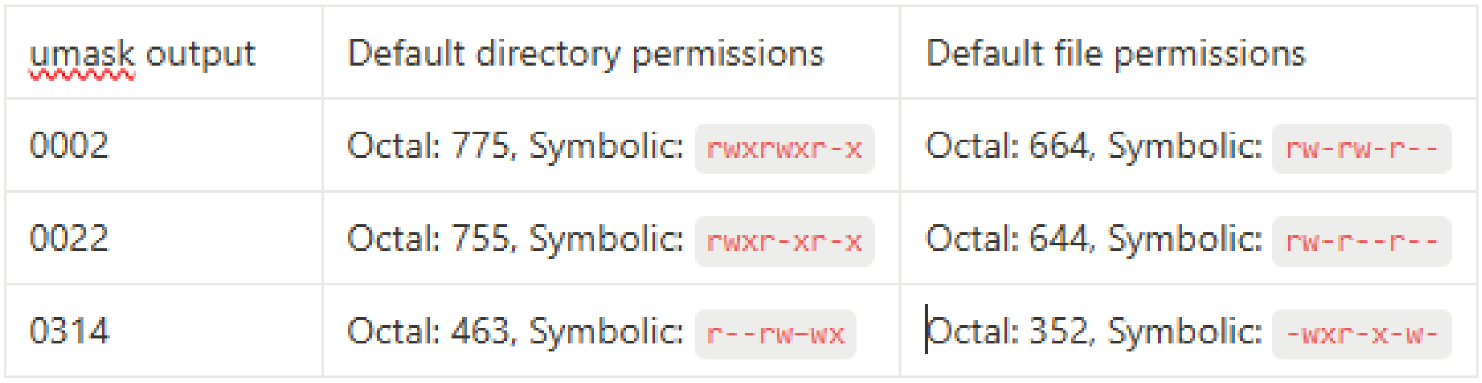
Here are some examples of chmod usage with octal notation:

Command in octal notation	Change in user (u) permissions	Change in group (g) permissions	Change in world (o) permissions
`chmod 777 deploy.sh`	✓ Read, ✓ Write, ✓ Execute	✓ Read, ✓ Write, ✓ Execute	✓ Read, ✓ Write, ✓ Execute
`chmod 501 deploy.sh`	✓ Read, ☐ Write, ✓ Execute	☐ Read, ☐ Write, ☐ Execute	☐ Read, ☐ Write, ✓ Execute
`chmod 365 deploy.sh`	☐ Read, ✓ Write, ✓ Execute	✓ Read, ✓ Write, ☐ Execute	✓ Read, ☐ Write, ✓ Execute
`chmod 177 backup.sh`	☐ Read, ☐ Write, ✓ Execute	✓ Read, ✓ Write, ✓ Execute	✓ Read, ✓ Write, ✓ Execute

The umask command allows you to check the default permissions for new files or directories.

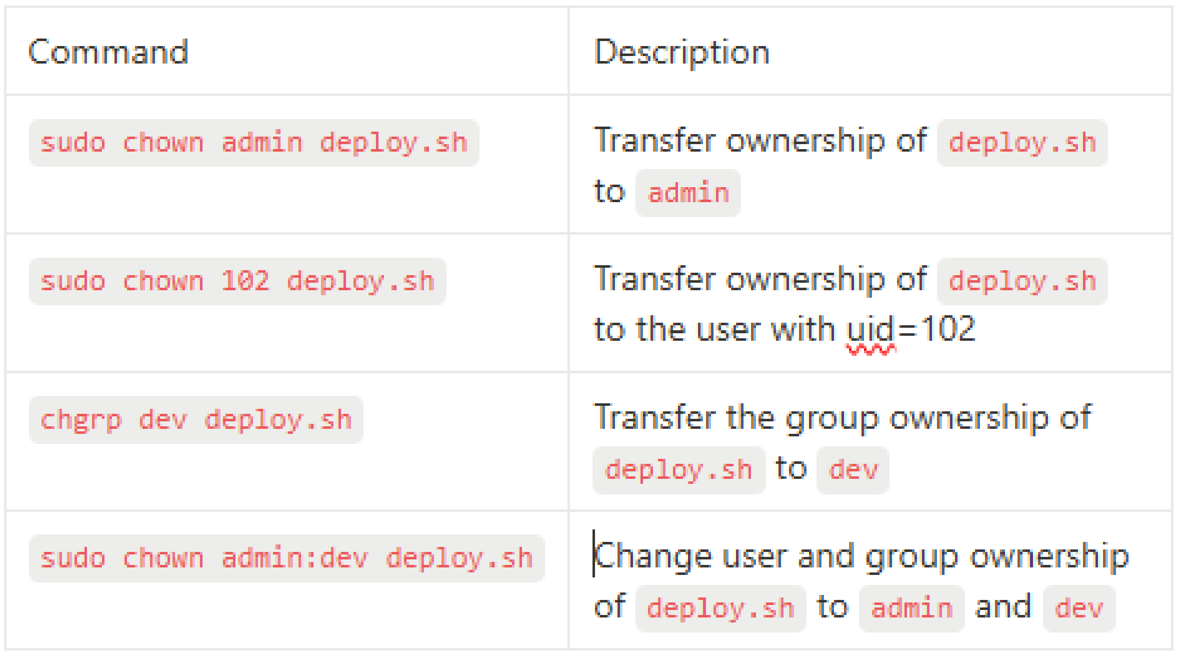
```
Command Description
umask Displays the default user and group permissions when
creating files/directories
```

Examples of umask output:



umask output	Default directory permissions	Default file permissions
0002	Octal: 775, Symbolic: `rwxrwxr-x`	Octal: 664, Symbolic: `rw-rw-r--`
0022	Octal: 755, Symbolic: `rwxr-xr-x`	Octal: 644, Symbolic: `rw-r--r--`
0314	Octal: 463, Symbolic: `r--rw-wx`	Octal: 352, Symbolic: `-wxr-x-w-`

To change the owner of a file or directory, you can use chown and chgrp. Here's how:



Command	Description
`sudo chown admin deploy.sh`	Transfer ownership of `deploy.sh` to `admin`
`sudo chown 102 deploy.sh`	Transfer ownership of `deploy.sh` to the user with uid=102
`chgrp dev deploy.sh`	Transfer the group ownership of `deploy.sh` to `dev`
`sudo chown admin:dev deploy.sh`	Change user and group ownership of `deploy.sh` to `admin` and `dev`

To perform tasks that require administrative access, use su or sudo:

```
Command Description
$ su Open superuser shell (root)
$ sudo deploy.sh Run deploy.sh with superuser privileges
$ sudo -i Open superuser shell if su is disabled
```

SSH Protocol

SSH (Secure Shell Protocol) is used to securely send commands to a computer over an insecure network.

It can be compared to a store owner giving instructions to an employee while traveling, where encryption ensures that no one else can overhear the conversation.
SSH uses cryptography for both authentication and encryption, making it a secure way to transmit data.
SSH utilizes tunneling to send data that might be blocked by network restrictions, such as firewalls.
Older protocols like Telnet were replaced by SSH because Telnet did not encrypt connections, leaving data vulnerable.

For encryption and authentication:
SSH uses public key cryptography, which involves two keys:
Public key: Used to decrypt the data.
Private key: Used to encrypt the data.

Both the client and server have public keys, but only the client holds the private key.
After the initial connection, a shared symmetric key is used for further encryption and decryption of data like passwords and files.
Symmetric keys are preferred because they are faster for encryption and decryption compared to asymmetric keys, which involve more complex algorithms.
The symmetric key is securely exchanged between the client and server at the start of the connection.
Unlike HTTPS, where only the server proves its identity through an SSL certificate, in SSH, both the client and server authenticate each other using their public keys.

SSH is used to securely connect to remote servers from local machines.
It is an essential tool for managing infrastructure and sending files over a network.
SSH is especially valuable for transferring files across networks with firewalls or other restrictions, ensuring data protection during transit

SSH Keys

SSH keys are a pair of cryptographic keys used to authenticate and secure connections between a client (computer) and a server.
SSH uses two keys in the key pair:

Public Key: This is shared with others (such as the server you're connecting to). It can be freely distributed and is used to decrypt the challenge during the SSH connection establishment.

Private Key: This is kept secret and stored securely on the client machine (your local computer). It is never shared and is used to decrypt the challenge and encrypt the challenge sent by the server during the connection establishment.

Public-Private keys are used only for initial connection setup and authentication, but the actual data transfer and encryption of data happen through a symmetric key that is shared between the client and server after authentication is successfully completed.

How SSH keys work:
Generate SSH keys on the client-side.
Copy the public key to the server.
Connect to the server from the client using SSH.
Disable password login on the server to enforce key-based authentication.

Why use SSH keys:
Passwordless login, making it more secure and convenient.
Avoid brute-force attacks, as SSH keys are much harder to crack than passwords.
Automate tasks by using key-based authentication for scripts and processes.

SFTP (Secure File Transfer Protocol):

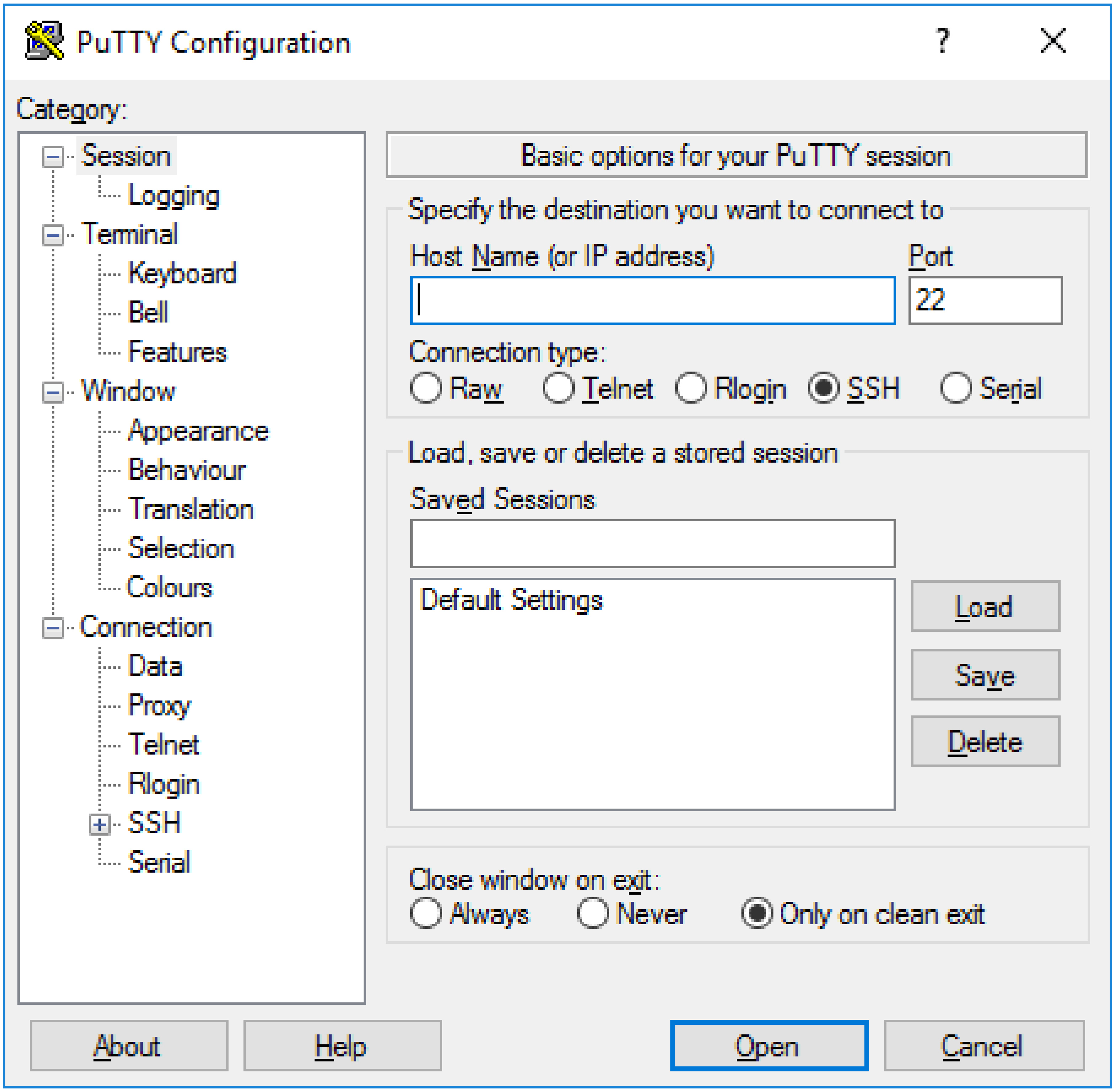
SFTP is a way to send files securely from a local computer to a remote server. Unlike FTP, it uses SSH to encrypt the data that needs to be transferred over the network.
Unlike FTP, which only supports password-based login, SFTP supports public key authentication for passwordless login.

SFTP is cross-platform, which means it can be used across different operating systems like Windows, Linux, and macOS.
SFTP can compress data, allowing it to use low bandwidth to send large files efficiently.
Popular SFTP client-side GUIs include FileZilla, WinSCP, and CyberDuck.

Why use SFTP:
FTP does not encrypt data during transfer, leaving it vulnerable. SFTP ensures encrypted data due to its use of SSH.
FTP only supports password-based login, but SFTP provides more secure logins like public key authentication.
SFTP allows for efficient file transfer, especially for large files over a low-bandwidth connection.

```
Create an SFTP Server:

- Install OpenSSH Server:
sudo apt-get install openssh-server

- Open the SSH Server Configuration File:
sudo nano /etc/ssh/sshd_config

- Add the following line if it's not already available:
subsystem sftp /usr/lib/openssh/sftp-server

- Start or Restart the SSH Server:
sudo service ssh start


2. Connect to an SFTP Client:

- You can either use password-based login or SSH public key
authentication.
For public key authentication, add your private key to the
client.
```

Windows SSH Client


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]