**Ubuntu: User Management and Linux Permissions**

**User Management in Ubuntu**

User management is a critical part of administering any Linux-based system like Ubuntu. This includes creating, managing, and deleting users and groups. Below are the steps and concepts related to user creation in Ubuntu.

**Creating a User in Ubuntu**

To create a new user on an Ubuntu system, you can use the adduser or useradd command:

1. **Using adduser Command:**
   * This is a more user-friendly command that interactsively asks for user details.
2. sudo adduser [username]

Example:

sudo adduser john

* + The system will prompt you to set a password and provide other user information like full name, room number, etc.

1. **Using useradd Command:**
   * This is a lower-level utility, and it does not prompt for additional details unless explicitly specified.
2. sudo useradd -m [username]
   * Use the -m option to create the user's home directory.
   * Example:
3. sudo useradd -m jane
4. sudo passwd jane
5. **Assigning User to a Group:** Users can be added to specific groups for permission management:
6. sudo usermod -aG [groupname] [username]

Example:

sudo usermod -aG sudo john

* + This command adds the user john to the sudo group, granting administrative privileges.

1. **Deleting a User:** If you need to delete a user, use the following commands:
2. sudo deluser [username]

Example:

sudo deluser john

* + To remove the user's home directory as well:

sudo deluser --remove-home [username]

**Linux Permissions**

File and directory permissions in Linux define the access level for users and groups. They are a cornerstone of Linux’s security model.

**Types of Permissions**

Linux permissions are divided into three categories:

1. **Read (r):** Permission to view the contents of a file or directory.
2. **Write (w):** Permission to modify the contents of a file or directory.
3. **Execute (x):** Permission to run a file (e.g., a script) or access a directory.

Each file or directory has three levels of access:

1. **Owner (u):** The user who owns the file.
2. **Group (g):** The group that the file is associated with.
3. **Others (o):** All other users on the system.

**Viewing Permissions**

To view file or directory permissions, use the ls -l command:

ls -l

Example output:

-rw-r--r-- 1 john users 4096 Jan 14 12:00 example.txt

Explanation:

* -rw-r--r-- represents the permissions.
  + The first character (‘-’) indicates the file type (‘d’ for directory).
  + The next nine characters are divided into three groups (owner, group, others).
    - rw- means the owner has read and write permissions.
    - r-- means the group has read-only permissions.
    - r-- means others have read-only permissions.

**Changing Permissions**

Use the chmod command to modify permissions.

1. **Symbolic Method:**
2. chmod [who][+/-][permission] [file]
   * Example:
   * chmod u+x example.txt # Adds execute permission for the owner
   * chmod g-w example.txt # Removes write permission for the group
   * chmod o+r example.txt # Adds read permission for others
3. **Octal Method:** Each permission is represented by a number:
   * Read (r) = 4
   * Write (w) = 2
   * Execute (x) = 1
   * No permission = 0

Combine these numbers to set permissions:

chmod 764 example.txt

* + Explanation:
    - Owner: 7 (read+write+execute)
    - Group: 6 (read+write)
    - Others: 4 (read-only)

**Changing Ownership**

The chown command changes the ownership of a file or directory.

1. **Change Owner:**
2. sudo chown [owner] [file]

Example:

sudo chown john example.txt

1. **Change Group:**
2. sudo chown :[group] [file]

Example:

sudo chown :developers example.txt

1. **Change Owner and Group:**
2. sudo chown [owner]:[group] [file]

Example:

sudo chown john:developers example.txt

**Advanced Permissions**

1. **Sticky Bit:**
   * Ensures that only the owner of a file can delete it from a directory.
2. chmod +t [directory]
   * Example:
   * chmod +t /shared-folder
3. **Setuid and Setgid:**
   * Setuid: Allows a program to run with the permissions of its owner.
   * Setgid: Allows a program or directory to run with the permissions of its group.
4. chmod u+s [file] # Setuid
5. chmod g+s [directory] # Setgid

By understanding and managing users and permissions effectively, you can ensure a secure and well-organized Linux environment.