Lecture 6: File and Directory Management Commands

Managing files and directories is a fundamental aspect of working with Linux. Below is a list of commonly used commands for file and directory management, along with examples for each:

1. `ls` - List directory contents

- Example: `ls -l`

- Lists files and directories in the current directory with detailed information (permissions, owner, size, and modification date).

2. `cd` - Change directory

- Example: `cd /home/user/Documents`

- Changes the current directory to `/home/user/Documents`.

3. `pwd` - Print working directory

- Example: `pwd`

- Displays the full path of the current working directory.

4. `mkdir` - Make directories

- Example: `mkdir new_folder`

- Creates a new directory named `new_folder` in the current directory.

- Example: `mkdir -p parent_folder/child_folder`

- Creates a directory `parent_folder` and a subdirectory `child_folder` in one command. The `-p` option ensures parent directories are created as needed.

5. `rmdir` - Remove empty directories

- Example: `rmdir old_folder`

- Deletes the directory `old_folder` if it is empty.

6. `rm` - Remove files or directories

- Example: `rm file.txt`

- Deletes the file `file.txt`.

- Example: `rm -r directory_name`

  - Recursively deletes the directory `directory_name` and all its contents. The `-r` option stands for "recursive."

- Example: `rm -f file.txt`

  - Forcibly deletes `file.txt` without prompting for confirmation. The `-f` option stands for "force."

7. `cp` - Copy files and directories

- Example: `cp file.txt /home/user/`

  - Copies `file.txt` to `/home/user/`.

- Example: `cp -r dir1/ dir2/`

  - Recursively copies the directory `dir1` to `dir2`. The `-r` option stands for "recursive."

8. `mv` - Move or rename files and directories

- Example: `mv old_name.txt new_name.txt`

  - Renames `old_name.txt` to `new_name.txt`.

- Example: `mv file.txt /home/user/`

  - Moves `file.txt` to `/home/user/`.

9. `touch` - Create empty files or update timestamps

- Example: `touch newfile.txt`

  - Creates an empty file named `newfile.txt` if it does not exist or updates its timestamp if it does.

10. `find` - Search for files and directories

- Example: `find /home/user/ -name "*.txt"`

  - Searches for all files with a `.txt` extension within the `/home/user/` directory and its subdirectories.

- Example: `find . -type d -name "folder_name"`

  - Searches for directories named `folder_name` in the current directory and its subdirectories.

11. `locate` - Find files by name
   - Example: `locate file.txt`
     - Searches for `file.txt` in the database of files maintained by `locate`.

12. `du` - Estimate file space usage
   - Example: `du -h`
     - Shows the disk usage of files and directories in human-readable format (e.g., KB, MB).
   - Example: `du -sh /home/user/`
     - Shows the total size of the directory `/home/user/` in a human-readable format.

13. `df` - Report file system disk space usage
   - Example: `df -h`
     - Displays disk space usage for all mounted filesystems in human-readable format.

14. `chmod` - Change file modes or Access Control Lists
   - Example: `chmod 755 script.sh`
     - Sets the permissions of `script.sh` to `rwxr-xr-x`.
   - Example: `chmod u+x script.sh`
     - Adds execute permission to the owner of `script.sh`.

15. `chown` - Change file owner and group
   - Example: `chown user:group file.txt`
     - Changes the ownership of `file.txt` to user `user` and group `group`.

16. `ln` - Create hard and symbolic links
   - Example: `ln -s /path/to/original /path/to/link`
     - Creates a symbolic (or soft) link named `/path/to/link` pointing to `/path/to/original`. The `-s` option stands for "symbolic."

17. `stat` - Display file or file system status

- Example: `stat file.txt`

  - Provides detailed information about `file.txt`, including size, permissions, and modification time.

These commands are the building blocks for navigating and managing files and directories in a Linux environment.

Managing users and groups is crucial for system administration in Linux. Below is a list of commands used for handling users and groups, along with examples for each:

User Commands

1. `adduser` / `useradd` - Add a new user

  - Example: `sudo adduser newuser`

   - Creates a new user named `newuser` and prompts for additional details such as password and user information.

  - Example: `sudo useradd -m -s /bin/bash newuser`

   - Creates a new user named `newuser` with a home directory (`-m`) and sets the default shell to `/bin/bash` (`-s`).

2. `usermod` - Modify a user account

  - Example: `sudo usermod -aG groupname username`

   - Adds `username` to `groupname`. The `-aG` option appends the user to the specified group without removing them from other groups.

  - Example: `sudo usermod -s /bin/zsh username`

   - Changes the default shell of `username` to `/bin/zsh`.

3. `deluser` / `userdel` - Delete a user

  - Example: `sudo deluser username`

   - Deletes the user `username` but keeps their home directory and files.

  - Example: `sudo userdel -r username`

   - Deletes the user `username` and their home directory along with their files (`-r`).

4. `passwd` - Change user password

  - Example: `sudo passwd username`

   - Prompts to enter a new password for `username`.

- Example: `passwd`
  - Changes the password for the currently logged-in user.

5. `whoami` - Display the current user
   - Example: `whoami`
     - Displays the username of the currently logged-in user.

6. `id` - Display user and group information
   - Example: `id username`
     - Shows user ID (UID), primary group ID (GID), and supplementary group IDs for `username`.
   - Example: `id`
     - Shows the UID and GID of the current user.

7. `groups` - Show the groups a user belongs to
   - Example: `groups username`
     - Lists the groups that `username` is a member of.
   - Example: `groups`
     - Lists the groups of the currently logged-in user.

Group Commands

1. `addgroup` / `groupadd` - Add a new group
   - Example: `sudo addgroup newgroup`
     - Creates a new group named `newgroup`.
   - Example: `sudo groupadd groupname`
     - Creates a new group `groupname`.

2. `delgroup` / `groupdel`- Delete a group
   - Example: `sudo delgroup oldgroup`
     - Deletes the group `oldgroup`.
   - Example: `sudo groupdel groupname`

- Deletes the group `groupname`. Note that the group must not have any members.

3. `groupmod` - Modify a group
   - Example: `sudo groupmod -n newgroupname oldgroupname`
     - Renames the group `oldgroupname` to `newgroupname`.

   - Example: `sudo groupmod -g 1001 newgroup`
     - Changes the GID of `newgroup` to `1001`.

4. `getent` - Get entries from administrative database
   - Example: `getent passwd username`
     - Displays the user information for `username` from the system database.

   - Example: `getent group groupname`
     - Displays information about `groupname`.

5. `gpasswd` - Administer `/etc/group` and `/etc/gpasswd` files
   - Example: `sudo gpasswd -a username groupname`
     - Adds `username` to the `groupname` group.

   - Example: `sudo gpasswd -d username groupname`
     - Removes `username` from `groupname`.

6. `newgrp` - Change the current group ID
   - Example: `newgrp groupname`
     - Changes the current group ID to `groupname` for the current session. This is useful when you want to switch group permissions for a current terminal session.

File and Directory Permissions Related to Users and Groups

1. `chmod` - Change file permissions
   - Example: `chmod 755 file.txt`

- Sets the file `file.txt` permissions to `rwxr-xr-x`.

2. `chown` - Change file owner and group
   - Example: `sudo chown username:groupname file.txt`
     - Changes the owner and group of `file.txt` to `username` and `groupname`.

   - Example: `sudo chown -R username:groupname /path/to/directory`
     - Recursively changes the ownership of all files and directories under `/path/to/directory`.

3. `chgrp` - Change group ownership
   - Example: `sudo chgrp groupname file.txt`
     - Changes the group ownership of `file.txt` to `groupname`.

These commands are essential for user and group management on a Linux system, allowing you to control access and permissions effectively.