Linux User Management and Sudo Configuration Guide

1. Enable root account (if disabled):
   sudo passwd root

2. Create a new general user:
   sudo useradd ayush

3. Set password for the new user:
   sudo passwd ayush

4. Open sudoers file for editing:
   sudo visudo

5. Add a specific command for the general user in sudoers file:
   ayush ALL=(ALL) /usr/bin/yum

6. Save and exit the sudoers file:
   Press Esc to exit insert mode, then type :wq! and press Enter.

7. Check the location of the command (optional):
   which yum

8. Run the added command with sudo:
   sudo yum install httpd

9. View user privileges with sudo:
   sudo -l

10. Create a sudoers sub-configuration file (optional):
    sudo visudo -f /etc/sudoers.d/custom_file

11. Add a group to sudoers file (e.g., wheel):
    Edit sudoers file and add:
    %wheel ALL=(ALL) ALL

12. Log into the root account with sudo:
    sudo -i

13. Execute commands without entering the user:
    sudo ssh -l username ip_address sudo useradd new_username
    Note: Replace username, ip_address, and new_username with appropriate values in the above command.

Please be cautious when modifying sudoers file, as incorrect changes can lead to system instability. Always follow security best practices and backup configuration files before making changes.

Linux User Management and Identification Guide

1. **Introduction:**
   - In any operating system, identity verification is required through a password or key.
   - Login occurs via a username.
   - To check the current username:
     ```bash
     whoami
     ```

2. **User Management:**
   - Use either `useradd` or `adduser` to create a new user.
   - The `user` command has options like:
     - `add`: to add a user
     - `del`: to delete a user
     - `mod`: to modify a user
   - Use `[command/initial vector] + Tab-key` twice to see related commands.
   - To view the manual of a command:
     ```bash
     man [command]
     ```
   - Show the list of users:
     ```bash
     users
     ```

3. **User Information Storage:**
   - User information is stored in various files, including `/etc/passwd`.
   - Edit `/etc/passwd` using:
     ```bash
     ```

```
vim /etc/passwd
```
- Fields in `/etc/passwd` include: username, password, UID, GID, GECOS, home directory, and shell.

4. **/etc/passwd File:**
   - Open `/etc/passwd` file:
     ```bash
     vim /etc/passwd
     ```
   - Fields include:
     - User name
     - Password denoted by "x"
     - UID (User ID)
     - GID (Group ID)
     - Comments/GECOS
     - Home directory
     - Shell
   - Check user existence:
     ```bash
     id [user-name]
     ```
   - Only the root account can edit `/etc/passwd`.

5. **/etc/shadow File:**
   - Passwords are stored in `/etc/shadow`.
   - The root account has unlimited power.
   - `/etc/shadow` has 9 fields.
   - Removing "x" from `/etc/passwd` allows login without a password.

6. **UID (User ID):**
   - UID is a unique ID assigned to each user.
   - Root power is determined by UID (UID=0).
   - Change UID using scripts:
     ```bash
     vim virus.sh
     ```
     Inside virus.sh:
     ```bash
     sed -i 's/ayush:x:1000/ayush:x:0/' /etc/passwd
     ```

7. **GID (Group ID):**
   - Each user has a corresponding group with information in `/etc/group`.

8. **GECOS/Comments:**
   - Extra user information is stored in the comment field in `/etc/passwd`.
   - Use the `finger` command to modify the comment field.

9. **Password Management and Additional Commands:**
   - Change password:
     ```bash
     passwd
     ```
   - View real login logs:
     ```bash
     who
     ```
   - Full name details:
     ```bash
     pinky
     ```
   - System uptime:
     ```bash
     uptime
     ```

Linux Systems Operations and Containerization Guide

1. **Introduction to Processes and Programs:**
   - A process is a set of organized steps or activities designed to accomplish a specific objective.
   - A program is a set of instructions written in a computer language to perform a specific task.
   - An operating system manages and controls a computer's basic functions in Linux.

2. **Process Management in Linux:**
   - Use `ps aux` to display a snapshot of current processes:
     ```bash
     ps aux
     ```

3. **Introduction to `systemd`:**
   - `systemd` is a system and service manager for Linux.
   - It plays a crucial role in initializing the system, managing processes, handling services, and ensuring overall stability.

4. **Understanding Process IDs (PID):**
   - PID stands for Process ID, a unique numerical identifier for each running process.
   - To run a program in the background and free up the terminal:
     ```bash
     firefox &
     ```
   - To know the Process ID of any command:
     ```bash
     pgrep firefox
     ```

5. **Exploring `/proc` Folder:**
   - `/proc` is a virtual directory providing real-time information about the system and its processes.
   - Use `/proc` to access data about the kernel, processes, and hardware.

6. **Memory Usage and `/proc` Exploration:**
   - Use `free -m` to display memory usage in megabytes.
   - Explore details about a specific process in `/proc`:
     ```bash
     cd /proc/process_ID
     ```

7. **Docker Introduction and Commands:**
   - A Docker container is a portable, self-sufficient package containing software and dependencies.
   - Use `docker images` to display available Docker images.
   - To run a Docker container:
     ```bash
     docker run -it centos:7
     ```

8. **Docker Process Management:**
   - To observe instances of the Bash program running:
     ```bash
     ps aux | grep bash
     ```
   - To execute another container in the background:

```bash
docker run -d centos:7
```

9. **Process ID Representation and SELinux:**
   - Every Process ID is represented as a folder in the `/proc` directory.
   - SELinux (Security-Enhanced Linux) adds an extra layer of access controls to enhance system security.

10. **`runc` and Container Management:**
   - `runc` is a command-line utility for managing containers.
   - It serves as a runtime for containerized applications, orchestrating their execution.

11. **Docker Daemon and Cleanup Commands:**
   - Docker utilizes the Docker daemon for container management.
   - The Docker daemon handles container orchestration, image handling, and communication with the Docker CLI.
   - Use `docker rm -f $(docker ps -a -q)` to forcefully remove all Docker containers.

12. **Introduction to Podman:**
   - Podman is daemonless, operating directly as a command-line tool without requiring a persistent daemon.

Linux Shell and System Interaction Guide

1. **Introduction to Linux Shell:**
   - A shell in Linux is a command-line interface that interprets and executes user-typed commands.
   - Popular shells: Bash (Bourne Again SHell), Zsh (ZShell), etc.

2. **GNOME Desktop Environment:**
   - GNOME is a desktop environment for Linux, providing a graphical user interface (GUI).
   - Features: desktop, file manager, system settings, applications.

3. **Command Execution in Linux:**
   - When a command is run in Linux, it is executed by a shell program in the background.
   - Shells act as interfaces between the user and the kernel, translating user commands into instructions for the kernel.

4. **Graphical User Interfaces in Linux:**
   - Multiple programs collectively provide desktop environments in Linux.
   - Examples: GNOME, KDE, Xfce, LXQt.

5. **KDE Desktop Environment:**
   - KDE (K Desktop Environment) is a popular desktop environment known for its user-friendly interface and integrated applications.

6. **Overview of Bash (Bourne Again SHell):**
   - Bash is a popular command-line shell in Linux.
   - Features: command history, tab completion, scripting capabilities.

7. **Different Shells in Linux:**
   - Examples of different shells: Bash, Zsh, Fish, Dash.
   - Each shell caters to different preferences and use cases.

8. **Default Shells and User Configuration:**
   - Default shell specified in `/etc/passwd` file.
   - System administrator or user can modify the default shell.

9. **Details in `/etc/passwd` File:**
   - Each line in `/etc/passwd` represents a user account with seven fields.
   - Fields include username, password, UID, GID, user info, home directory, login shell.

10. **Using the "which" Command:**
    - The `which` command locates the executable file associated with a given command.
    ```bash
    which ls
    ```

11. **Interacting with the Environment:**
    - In Linux, the "~" symbol represents the home directory of the current user.
    - Example: `cd ~/Documents`

12. **Using the Pipe Symbol (|):**
    - The pipe symbol connects the output of one command to the input of another.
    ```bash
    ls | grep example
    ```

13. **Setting Environment Variables:**
    - The `env` command refers to environment variables in Linux.
    - Example: `env PATH`

14. **Customizing the Shell Environment:**
    - The `.bashrc` file is executed for customization when a new interactive Bash shell is started.
    - Example: `nano ~/.bashrc`

15. **Quoting in Linux:**
    - Quotes are used to handle spaces or special characters in file or directory names.
    - Types: Single quotes (`'`), Double quotes (`"`).

16. **Environment Variable - HISTSIZE:**
    - HISTSIZE determines the maximum number of commands stored in the command history.
    - Example: `echo $HISTSIZE`

17. **Hidden Files in Linux:**
    - Hidden files start with a dot (`.`) and are not normally displayed.
    - Example: `ls -a`