AlMustafa University

Jamiat AlMustafa

Linux System Administration

Lecture 3

Managing System Administration Tasks

Instructor: Dr. Husam Alkinani

Ph.D. in Computer Science and Engineering

Linux System Administration Course

Last Updated: October 2025

Managing System Administration Tasks

Managing System Administration Tasks

Introduction to System Administration

System administration involves the comprehensive management of computer systems, networks, and services. It encompasses user management, system maintenance, security implementation, and resource optimization to ensure smooth operation of Linux systems.

User Management

□ User Management Definition

User management refers to controlling users on a Linux system, including creating new user accounts, modifying their permissions, and deleting them when necessary. It also involves assigning specific roles to users such as system administrator or regular user.

Why Manage Users?

1. Security Enhancement

- ▷ The primary goal of user management is to secure the system by controlling access and permissions

- ▷ Implementation through password management, permission control, and user group organization
- ▷ Prevention of unauthorized access to sensitive system resources
- ▷ Users can be categorized with specific permissions based on their roles and responsibilities

2. Resource Allocation

- ▷ Efficient distribution of system resources including memory, CPU, and storage
- ▷ Prevention of resource monopolization by individual users
- ▷ Maintenance of optimal system performance through balanced resource usage
- ▷ Implementation of quotas and limits to ensure fair resource distribution

3. Privacy Protection

- ▷ Creation of isolated user environments with private workspaces
- ▷ Protection of personal files and configurations from unauthorized access
- ▷ Implementation of user-specific security policies and access controls

4. Organization and Efficiency

- ▷ Systematic organization of users into functional groups
- ▷ Role-based access control for different user categories
- ▷ Streamlined administration through group-based permission management
- ▷ Enhanced workflow efficiency through proper user categorization

5. Auditing and Monitoring

- ▷ Comprehensive tracking of user activities and system access
- ▷ Detection of suspicious or anomalous user behavior
- ▷ Resource usage analysis and performance monitoring
- ▷ Compliance assurance with organizational security policies

6. Maintenance and Management

- ▷ Simplified user lifecycle management (creation, modification, deletion)

- ▷ Efficient permission updates based on changing organizational needs
- ▷ Automated user provisioning and de-provisioning processes
- ▷ Centralized user configuration management

7. Collaboration and Sharing

- ▷ Controlled file and resource sharing between users
- ▷ Implementation of collaborative workspaces with appropriate permissions
- ▷ Balance between collaboration needs and security requirements

What Happens When a New User is Created?

User Creation Process

When a new user is created in a Linux system, several automated processes ensure the user receives a secure, personalized, and functional environment. Understanding these processes is crucial for effective system administration.

1. **User Database Entry**: A new entry is added to `/etc/passwd` containing username, UID, GID, home directory path, and default shell information
2. **Home Directory Creation**: A dedicated home directory is established at `/home/username` providing personal workspace
3. **Skeleton Files Deployment**: Default configuration files from `/etc/skel` are copied to the user's home directory
4. **Permission Assignment**: Appropriate ownership and permissions are set on the home directory ensuring security and privacy
5. **Password Configuration**: Encrypted password entry is created in `/etc/shadow` for secure authentication
6. **Group Membership**: Primary group creation and assignment, with optional secondary group memberships
7. **Shell Assignment**: Default shell configuration, typically `/bin/bash` or `/bin/zsh`

Basic Commands for User Creation

Method A: Using adduser Command

□ Interactive User Creation

```
sudo adduser username
```

This command provides an interactive interface prompting for password and additional user information.

Method B: Using useradd Command

□ Manual User Creation

```
sudo useradd -m -s /bin/zsh username
sudo passwd username
```

□ Command Options Explanation

- ▷ **-m**: Creates the user's home directory automatically
- ▷ **-s**: Specifies the default shell for the user

Group Management Commands

Group Operations

```
# Create a new group
sudo groupadd groupname

# Add user to existing group
sudo usermod -aG groupname username

# Verify group membership
groups username
```

Group Management Warning

When using `usermod -aG`, the **-a** flag is crucial. Without it, the user will be removed from all other groups and only added to the specified group.

Command Comparison: adduser vs useradd

Aspect	**adduser**	**useradd**
Interactivity	Interactive prompts for user information	Non-interactive, requires manual configuration
Default Settings	Automatic home directory, shell, and configuration setup	Manual specification of all parameters required
Target Users	Beginners and standard administrative tasks	Advanced users requiring precise control
Automation	Limited automation capabilities	Better suited for scripted user creation

Table 1: Comparison of User Creation Commands

User Switching

Switch User Context

```
sudo su username
```

Note: The su command derives from ”switch user” and requires appropriate privileges.

The Shell Environment

Shell Definition

The Shell is a command-line interface program that serves as an intermediary between users and the operating system, enabling command execution and system interaction through text-based commands.

Zsh (Z Shell) Features

Zsh represents an advanced shell offering enhanced functionality beyond traditional shells:

- **Intelligent Auto-completion**: Advanced command and path completion with context awareness
- **Visual Enhancements**: Rich color support and customizable prompt formatting
- **Plugin Ecosystem**: Extensive plugin support through frameworks like Oh-My-Zsh
- **Improved Scripting**: Enhanced scripting capabilities with better syntax

Common Shell Types

1. **Bash (Bourne Again Shell)**: Most widespread shell in modern Linux distributions
2. **Fish (Friendly Interactive Shell)**: User-friendly shell with advanced interactive features
3. **Ksh (Korn Shell)**: Enhanced version of the original Bourne Shell
4. **Tcsh (TENEX C Shell)**: Advanced C Shell with additional functionality
5. **Dash (Debian Almquist Shell)**: Lightweight, fast shell for system scripts

6. **Csh (C Shell)**: Shell with C programming language-like syntax
7. **Sh (Bourne Shell)**: Original Unix shell, foundation for modern shells

System Maintenance

System Maintenance Overview

System maintenance involves regularly updating the system, ensuring that essential services are running correctly, monitoring resource usage (such as CPU, memory, and disk space), and fixing any potential issues that may arise.

Key Aspects of System Maintenance

1. Regular System Updates

▷ Keeping the system up-to-date with the latest security patches and software updates is crucial to maintain stability and security.

2. Ensuring Services Are Running Properly

▷ Check the status of essential services and make sure they are functioning as expected. This can include web servers, databases, or other background services.

3. Resource Monitoring

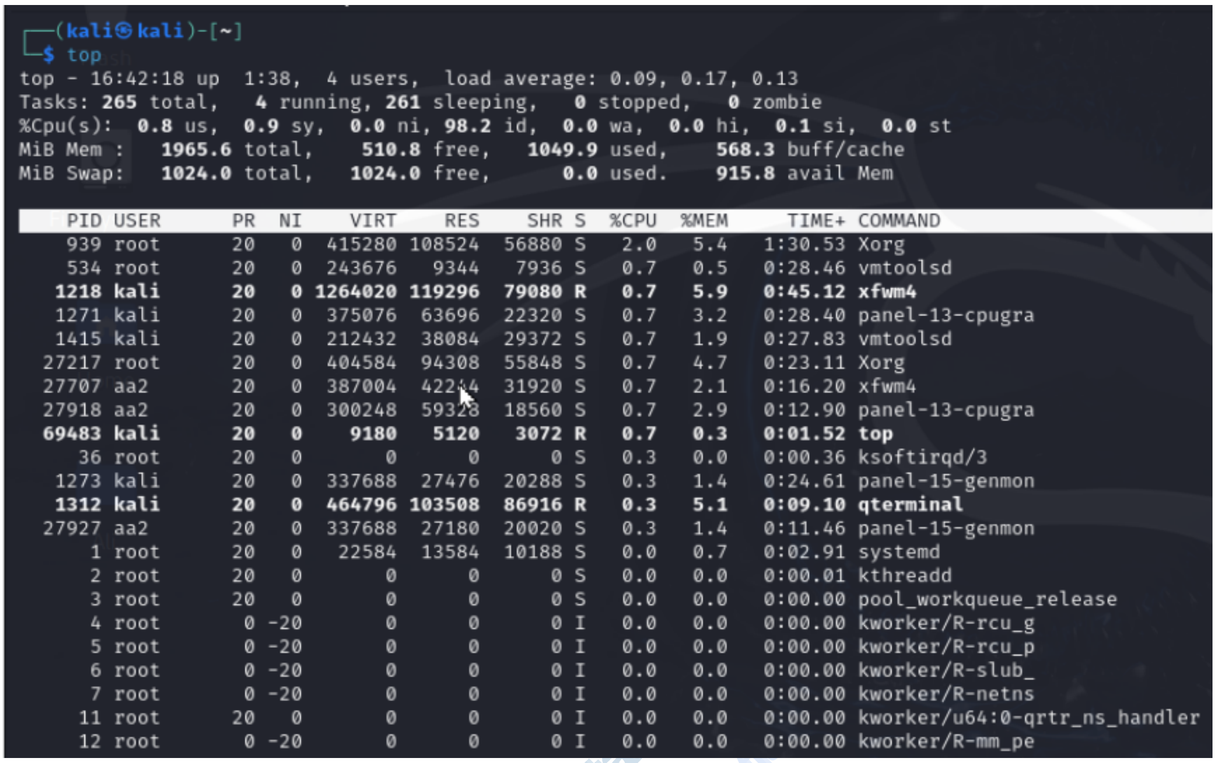
▷ Monitoring the system's resource consumption ensures that the CPU, memory, and disk space are being used efficiently. This helps to prevent system slowdowns or crashes due to resource exhaustion.

4. Fixing Issues

▷ When problems occur, it is important to troubleshoot and resolve them quickly to keep the system running smoothly.

top Command

top is a powerful tool for monitoring processes and resource usage on the system in real-time. It provides detailed information about the system's performance.



```
┌──(kali㉿kali)-[~]
└─$ top
top - 16:42:18 up  1:38,  4 users,  load average: 0.09, 0.17, 0.13
Tasks: 265 total,   4 running, 261 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.8 us,  0.9 sy,  0.0 ni, 98.2 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   1965.6 total,    510.8 free,   1049.9 used,    568.3 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.    915.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    939 root      20   0  415280 108524  56880 S   2.0   5.4   1:30.53 Xorg
    534 root      20   0  243676   9344   7936 S   0.7   0.5   0:28.46 vmtoolsd
   1218 kali      20   0 1264020 119296  79080 R   0.7   5.9   0:45.12 xfwm4
   1271 kali      20   0  375076  63696  22320 S   0.7   3.2   0:28.40 panel-13-cpugra
   1415 kali      20   0  212432  38084  29372 S   0.7   1.9   0:27.83 vmtoolsd
  27217 root      20   0  404584  94308  55848 S   0.7   4.7   0:23.11 Xorg
  27707 aa2       20   0  387004  42244  31920 S   0.7   2.1   0:16.20 xfwm4
  27918 aa2       20   0  300248  59328  18560 S   0.7   2.9   0:12.90 panel-13-cpugra
  69483 kali      20   0    9180   5120   3072 R   0.7   0.3   0:01.52 top
     36 root      20   0       0      0      0 S   0.3   0.0   0:00.36 ksoftirqd/3
   1273 kali      20   0  337688  27476  20288 S   0.3   1.4   0:24.61 panel-15-genmon
   1312 kali      20   0  464796 103508  86916 R   0.3   5.1   0:09.10 qterminal
  27927 aa2       20   0  337688  27180  20020 S   0.3   1.4   0:11.46 panel-15-genmon
      1 root      20   0   22584  13584  10188 S   0.0   0.7   0:02.91 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/u64:0-qrtr_ns_handler
     12 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_pe
```

Figure 1: Top Command Output Display

General Information at the Top of the Screen

1. **16:42:18**: The current system time.
2. **up 1:38**: The system uptime since the last reboot (1 hour and 38 minutes).
3. **4 users**: The number of users currently connected to the system.
4. **load average: 0.09, 0.17, 0.13**: The system's load average over the last 1, 5, and 15 minutes. These numbers represent the average number of processes waiting to be executed by the CPU.

Tasks Section

▷ **Tasks: 265 total**: Total number of processes on the system.

- **4 running**: The number of processes currently running.
- **261 sleeping**: The number of processes in a sleeping state.
- **0 stopped**: The number of stopped processes.
- **0 zombie**: The number of zombie processes (processes that have finished but are still using some resources).

CPU Usage

- **%Cpu(s): 0.8 us**: Percentage of CPU time spent on user processes.
- **0.9 sy**: Percentage of CPU time spent on system processes (kernel).
- **0.0 ni**: Percentage of CPU time spent on low-priority processes (nice value).
- **98.2 id**: Percentage of CPU time that the CPU is idle.
- **0.0 wa**: Percentage of CPU time waiting for I/O operations (disk, network).
- **0.0 hi**: Percentage of CPU time spent handling hardware interrupts.
- **0.1 si**: Percentage of CPU time spent handling software interrupts.
- **0.0 st**: Percentage of CPU time stolen by virtual machines (if the system is running in a virtualized environment).

Memory Usage

- **MiB Mem: 1965.6 total**: Total available system memory in megabytes.
- **510.8 free**: The amount of free, unused memory.
- **1049.9 used**: The amount of memory currently in use.
- **568.3 buff/cache**: Memory used by the system for caching and buffering.

Swap Usage

- **MiB Swap: 1024.0 total**: Total available swap space (virtual memory) in megabytes.
- **1024.0 free**: Amount of unused swap space.
- **0.0 used**: Amount of swap space currently in use.
- **915.8 avail Mem**: The available memory for immediate use, including reserved and currently used memory.

Process Details

In the top command output, each process running on the system is displayed with various attributes. Here's a breakdown of each column and its meaning:

Symbol	Description
PID	Process ID: The unique identification number assigned to the process.
USER	User: The user who initiated or is running the process.
PR	Priority: The priority level of the process.
NI	Nice value: The nice value, which determines the priority of the process.
VIRT	Virtual Memory: The total amount of virtual memory the process is using, which includes RAM and swap memory.
RES	Resident Memory: The amount of physical memory (RAM) the process is currently using.
SHR	Shared Memory: The amount of memory shared between different processes.
S	State: The current state of the process (-R Running,-S Sleeping,-Z Zombie,-T Stopped).
%CPU	The percentage of CPU usage that the process is consuming.
%MEM	The percentage of the total system memory (RAM) that the process is using.
TIME+	The total CPU time consumed by the process since it started.
COMMAND	The name of the command or program running the process.

Table 2: Process Information Fields in top Command

System Service Management

Service Management Overview

Managing system services is crucial for ensuring that all necessary services are running correctly and efficiently. This includes starting, stopping, restarting services, and checking their status.

Common Service Management Commands

Service Operations

```
# Restart a service
sudo systemctl restart service_name

# Start a service
sudo systemctl start service_name

# Stop a service
sudo systemctl stop service_name

# Check the status of a service
sudo systemctl status service_name

# Enable a service to start on boot
sudo systemctl enable service_name
```

Granting Administrative Privileges

To grant a user administrative (sudo) privileges so they can execute administrative commands like `sudo systemctl enable apache2` or any other commands requiring root access, you need to add the user to the sudo group (or the appropriate group that grants these privileges depending on your distribution).

Granting Sudo Privileges

```
# Add the user to the sudo group
sudo usermod -aG sudo username

# Verify the user has been added to the sudo group
groups username
```

Note: After adding the user to the sudo group, the user must log out and log back in for the changes to take effect.

Changing User Attributes

User Modification Commands

```
# Change the username
sudo usermod -l new_username old_username

# Change the home directory and move contents
sudo usermod -d /home/new_username -m new_username

# Change the password
sudo passwd username
```

Deleting a User

User Deletion Commands

```
# Delete a user
sudo deluser username

# Delete a user and their home directory
sudo deluser --remove-home username
```

Task Scheduling

Task Scheduling Overview

Task scheduling in Linux is typically done using the cron tool, which is a powerful utility used to run commands or programs on a scheduled basis at specified times. It automatically checks every minute when activated.

Note: The word ”cron” is derived from the Greek word ”chronos” meaning time.

Checking Cron Status

To check if cron is activated:

Check Cron Service

```
sudo systemctl status cron
```

▷ If you see a message like:
`Active: active (running) since Fri 2024-10-11 22:42:59 +03; 12h ago`
This means cron is active. If it’s inactive, you can activate it using the following commands:

Activate Cron Service

```
sudo systemctl enable cron
sudo systemctl start cron
```

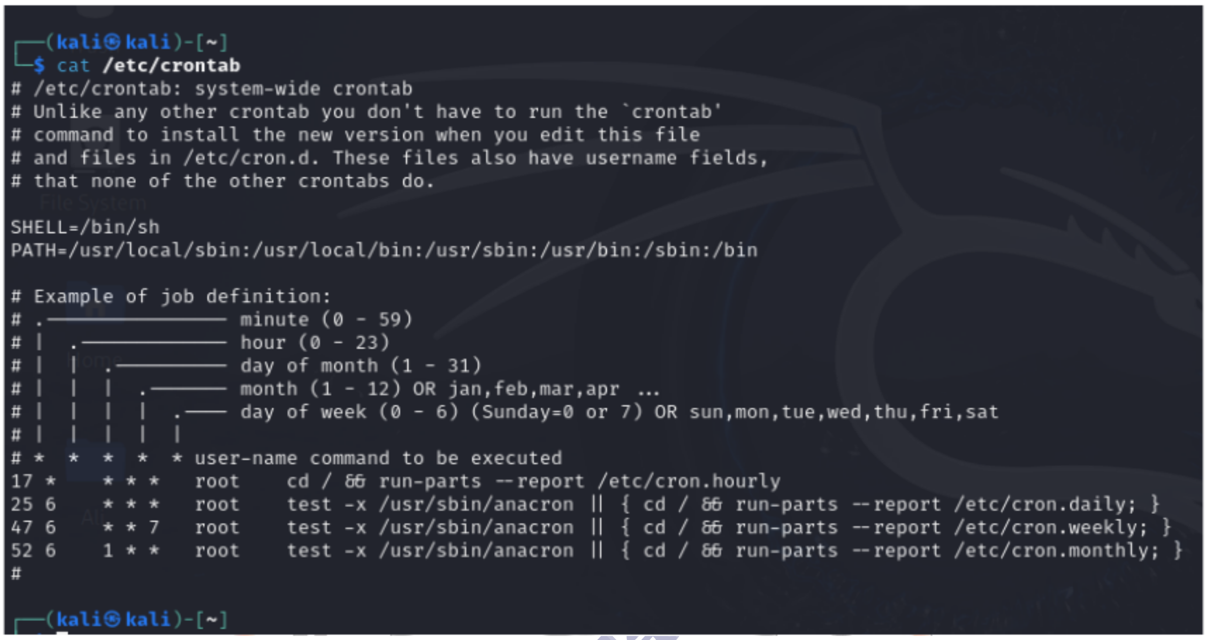
Crontab (Task Scheduler Table)

All scheduled tasks are stored in a table known as crontab. To understand how cron works:

View Crontab Content

```
cat /etc/crontab
```

This command shows the structure and the scheduled tasks in the crontab file.



```
┌──(kali㉿kali)-[~]
└─$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6    * * 7   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6    1 * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
#

┌──(kali㉿kali)-[~]
```

Figure 2: Crontab Command Structure

```
* * * * * command_to_be_executed
```

The structure of a cron job consists of five time and date fields, followed by the command to be executed. Here's what each field represents:

- Minute (0 - 59)
- Hour (0 - 23)
- Day of the month (1 - 31)
- Month (1 - 12)
- Day of the week (0 - 7), where 0 and 7 represent Sunday.

Examples of Cron Job Scheduling

1. To run a task every minute: `* * * * * command_to_be_executed`
2. To run a task daily at 2:00 AM: `0 2 * * * command_to_be_executed`
3. To run a task every Monday at 3:30 PM: `30 15 * * 1 command_to_be_executed`
4. To run a task every 15 minutes: `*/15 * * * * command_to_be_executed`

How to Schedule Tasks Using Cron

1. To add or modify the current user’s scheduled tasks:

Edit User Crontab

```
crontab -e
```

▷ The **-e** option opens a text editor to input scheduled tasks.

2. To display the scheduled tasks:

List User Crontab

```
crontab -l
```

▷ The **-l** option lists all the scheduled tasks for the current user.

3. To remove all scheduled tasks:

Remove User Crontab

```
crontab -r
```

▷ The **-r** option removes all the scheduled tasks for the current user.

Scheduling Examples

Creating folders using cron:

1. Create folder after one minute:

Cron: 1 Minute

```
* * * * * mkdir /home/kali/Desktop/BB1
```

2. Create folder after two minutes:

Cron: 2 Minutes

```
*/2 * * * * mkdir /home/kali/Desktop/BB1
```

3. Create folder after 1.5 minutes:

Cron: 1.5 Minutes

```
* * * * * sleep 30 && mkdir /home/kali/Desktop/BB1
```

Scheduling One-Time Tasks Using at

1. Install at if needed:

You may need to install at on your system if it's not already available.

2. Running a task at a specific time:

Create folder at specific time:

At Command

```
at 7:30 PM
mkdir /home/kali/Desktop/BB12
# Press Ctrl+D to finish
```

3. To view scheduled but not yet executed tasks:

View Pending Tasks

```
atq
```

This command lists pending tasks in the queue.

4. To remove a pending task:

Remove Pending Task

```
atrm job_number
```

Replace job_number with the task number from the atq list.

More Examples with at

Schedule a task for a specific date and time:

Schedule Task for Specific Date

```
at 10:00 AM Dec 15
```

Schedule a task to run after a specific duration:

Run task in 2 days:

At: Future Duration

```
at now + 2 days
```

Use other time formats:

Use relative time formats:

At: Relative Time

```
at 9:00 AM tomorrow
at 10:00 AM next week
```

Using at with echo to schedule via piping:

Schedule Task via Echo

```
echo "mkdir /home/kali/Desktop/BB12" | at 1:00 PM tomorrow
```

Software and Package Management in Linux

Package Management Overview

Package management in Linux is a key concept that allows users to install, update, and remove software efficiently. Each Linux distribution has its own package management system, which organizes and simplifies the process of installing, updating, and removing software.

Importance of Package Management

- ▷ **Easy software installation**: The package management system provides a way to install software along with all its necessary dependencies.
- ▷ **Software and system updates**: Programs can be updated easily using the package manager, improving security and performance.
- ▷ **Program removal**: Package management makes it easy to safely remove programs without leaving unnecessary files on the system.
- ▷ **Dependency management**: The package manager takes care of installing all the libraries and files a program needs to work correctly.

Types of Package Management Systems

There are many package management systems, and each Linux distribution has its preferred system. Some of the main package management systems include:

1. **APT (Advanced Packaging Tool)**: Used in Debian-based distributions like Ubuntu.
2. **YUM (Yellowdog Updater, Modified) and DNF**: Used in Red Hat-based distributions like CentOS and Fedora.
3. **Pacman**: Used in Arch-based distributions like Arch Linux.
4. **Zypper**: Used in SUSE Linux distributions.

Package Sources

Packages are usually stored in repositories managed by the community or official development teams of the distribution. From these repositories, users can download and update software safely.

Repositories: Internet repositories are considered the official and trusted sources for packages and are defined in the `/etc/apt/sources.list` file.

Package Management Commands

Updating repositories

Update Package Repositories

```
sudo apt update
```

Upgrading packages

Update all installed software:

APT Upgrade

```
sudo apt upgrade
```

Installing software

Install text editor Vim:

APT Install

```
sudo apt install vim
```

Removing software

Remove Software Package

```
sudo apt remove vim
```

This removes the program but keeps the personal configuration files.

Completely removing software (with configuration files)

Purge Software Package

```
sudo apt purge vim
```

Searching for packages

Search for Package

```
apt search package_name
```

Example:

Search for Specific Package

```
apt search nmap
```

Cleaning up the system

After removing or updating packages, you can clean up unnecessary files using the command:

Clean Up Unused Packages

```
sudo apt autoremove
```

Installing a local Debian package (.deb)

Installing a local Debian package (.deb) on a Debian-based system like Ubuntu or Kali Linux:

Install Local Debian Package

```
sudo dpkg -i package_name.deb
```

Sometimes the system may have dependency issues (missing libraries or programs). Fix dependency issues:

Fix Dependencies

```
sudo apt --fix-broken install
```

Installing and upgrading a specific package

Update Specific Package

```
sudo apt update && sudo apt upgrade nmap
```

Snap

Snap is a package management system developed by Canonical (the company behind Ubuntu). Snap allows you to install and manage software on a wide range of Linux distributions. Snap packages come with all necessary dependencies and run in isolation from the main system (sandboxing).

To install Snap

Install Snap Package Manager

```
sudo apt install snapd
sudo systemctl enable --now snapd
sudo systemctl status snapd
sudo systemctl restart snapd
```

Install programs with Snap:

Snap Install

```
sudo snap install nmap
```

H.W

1. Explain the importance of user management in a Linux system and how it enhances system security and ensures user privacy. In your answer, include how passwords, permissions, and user groups are managed.

2. What is the difference between the adduser and useradd commands when creating a new user in Linux? Explain when it is preferable to use each. Discuss the aspects of interactivity and default settings for both commands.

3. How is the environment for a new user customized in Linux? Discuss the key steps that occur when a new user is created, including the setup of the home directory and the assignment of the default shell.

4. Explain how system resources like memory and CPU are allocated to users in a Linux system. How does this contribute to maintaining system performance and ensuring fair resource distribution?

5. Discuss the role of user management in facilitating collaboration between users on a Linux system. How are permissions organized to enable file and folder sharing between users while maintaining security?