Managing System Administration Tasks:

User Management: This refers to controlling users on a Linux system, including creating new user accounts, modifying their permissions, and deleting them if necessary. You can also assign specific roles to users (like system administrator or regular user).

Why Manage Users?

1. **Security:**
   - ❖ The main goal of user management is to secure the system by controlling who has access and what permissions they have. This is done through managing passwords, permissions, and user groups.
   - ❖ Users can be grouped into categories with specific permissions, which prevents unauthorized access to sensitive resources.
2. **Resource Allocation:**
   - ❖ User management allows for the allocation of resources like memory, CPU, and storage to each user. This helps maintain system performance and prevents one user from using too many resources.
3. **Privacy:**
   - ❖ By creating separate user accounts, the system provides each user with a private workspace, ensuring their files and settings are protected from unauthorized access by other users.
4. **Organization and Efficiency:**
   - ❖ It makes system management easier by organizing users into groups based on their tasks or roles. For example, you can create groups for developers, administrators, or regular users, each with specific access permissions to certain parts of the system.
5. **Auditing and Monitoring:**
   - ❖ User management allows for tracking the activities of each user. This helps monitor suspicious or unusual behavior, analyze resource usage, and ensure compliance with security policies.
6. **Easy Maintenance and Management:**
   - o When managing many users, user management becomes essential for ensuring easy maintenance and updates. You can easily add, remove, or modify user permissions based on the needs of the organization.
7. **Collaboration and Sharing:**
   - ❖ Different users can collaborate on the system by sharing files and folders in a controlled way. You can specify who has permission to read, write, or modify files.

When a new user is created in a Linux system, several changes and configurations happen to ensure that the new user has an independent and secure working environment. Here's what happens in detail:

- **A new entry is added to the /etc/passwd file**, which stores basic information about each user. This entry includes the username, UID (User ID), GID (Primary Group ID), and information about the user's home directory and shell (the command-line interface used by the user).
- **A new home directory is created** for the user inside /home/username, where "username" is the name of the new user. This directory contains personal configuration files for the user, allowing them to have a customized environment.
- **Default files are copied from the "skeleton" directory (/etc/skel)** to the new user's home directory. These are initial configuration files that the user can modify as needed.
- **Permissions are set on the home directory** to ensure that the new user is the owner, usually with read, write, and execute permissions only for the user, ensuring privacy and security.
- **A password is assigned to the new user**, and it is stored in an encrypted format in the **/etc/shadow** file, which is used to securely manage user passwords.
- **A new group with the same name as the username is created**, and the user is usually the sole member of this group. The user can also be added to other groups depending on the system's requirements or the user's role.
- **A default shell is assigned to the user**, typically **/bin/bash** or **/bin/zsh**, which is the environment the user will use to run commands and interact with the system.

These steps ensure that each user on the Linux system has a secure, personalized environment with proper permissions and configurations.

Basic Commands for Creating a User:

A. **sudo adduser username**

After running this command, you will be prompted to set a password and provide some additional optional information.

B. **sudo useradd -m -s /bin/zsh username**

- ❖ -m: Ensures that the home directory for the user is created.
- ❖ -s: Specifies the default shell (in this case, /bin/zsh).

- **sudo passwd username** This command sets the password for the newly created user.

- **sudo usermod -aG groupname username**
    - ❖ -aG: This option means "append to Group". It adds the user to the specified group without removing them from other groups.
    - ❖ If you use -G without -a, the user will be removed from all other groups and only added to the specified group.

Note: Before adding a user to a group, the group must already exist. You can create it with the following command:

- sudo groupadd groupname   This command creates a new group.
- groups groupname   This command is used to verify that the group exists.

Difference Between the Two Commands:

- **Interactivity:**
    - ❖ adduser: It is interactive, prompting you to enter additional information automatically.
    - ❖ useradd: Non-interactive, requiring manual settings.
- **Default Settings:**
    - ❖ adduser: Automatically creates the home directory, assigns a default shell, and sets other configurations.
    - ❖ useradd: Does not perform these tasks unless you include the appropriate options (e.g., -m to create the home directory).
- **Target User:**
    - ❖ adduser: Suitable for users who prefer a simple and automatic process.
    - ❖ useradd: Intended for more advanced users who want greater control over the user creation process.

NOTE: If you have sudo privileges, you can switch to the user with the following command:

sudo su username   su from switch user.

The Shell is a program that acts as an interface between the user and the operating system, allowing users to interact with the system and execute commands.

Zsh (Z Shell): is one of the most popular shells in Linux systems. It's a command-line interface that allows you to execute commands and interact with the system.

Zsh stands out from other shells due to several additional features that make it popular among advanced users, such as:

- ❖ **Smart Autocompletion**: It provides faster and more accurate command and directory completion.
- ❖ **Visual Enhancements**: It supports colors and advanced prompt formatting.

Common Types of Shells:

1. **Bash (Bourne Again Shell)**: The most widely used shell in modern Linux distributions due to its simplicity.
2. **Fish (Friendly Interactive Shell)**: A highly interactive and user-friendly shell designed for ease of use and interactive features.
3. **Ksh (Korn Shell)**: One of the older shells, seen as an improvement over the original Bourne Shell.
4. **Tcsh (TENEX C Shell)**: A development of the original C Shell (csh), offering new features while keeping the same command style.
5. **Dash (Debian Almquist Shell)**: A lightweight and fast shell commonly used in Debian and its derivatives.
6. **Csh (C Shell)**: An older shell whose syntax is similar to the C programming language.
7. **Sh (Bourne Shell)**: One of the earliest shells developed, and it serves as the foundation for modern shells like Bash.

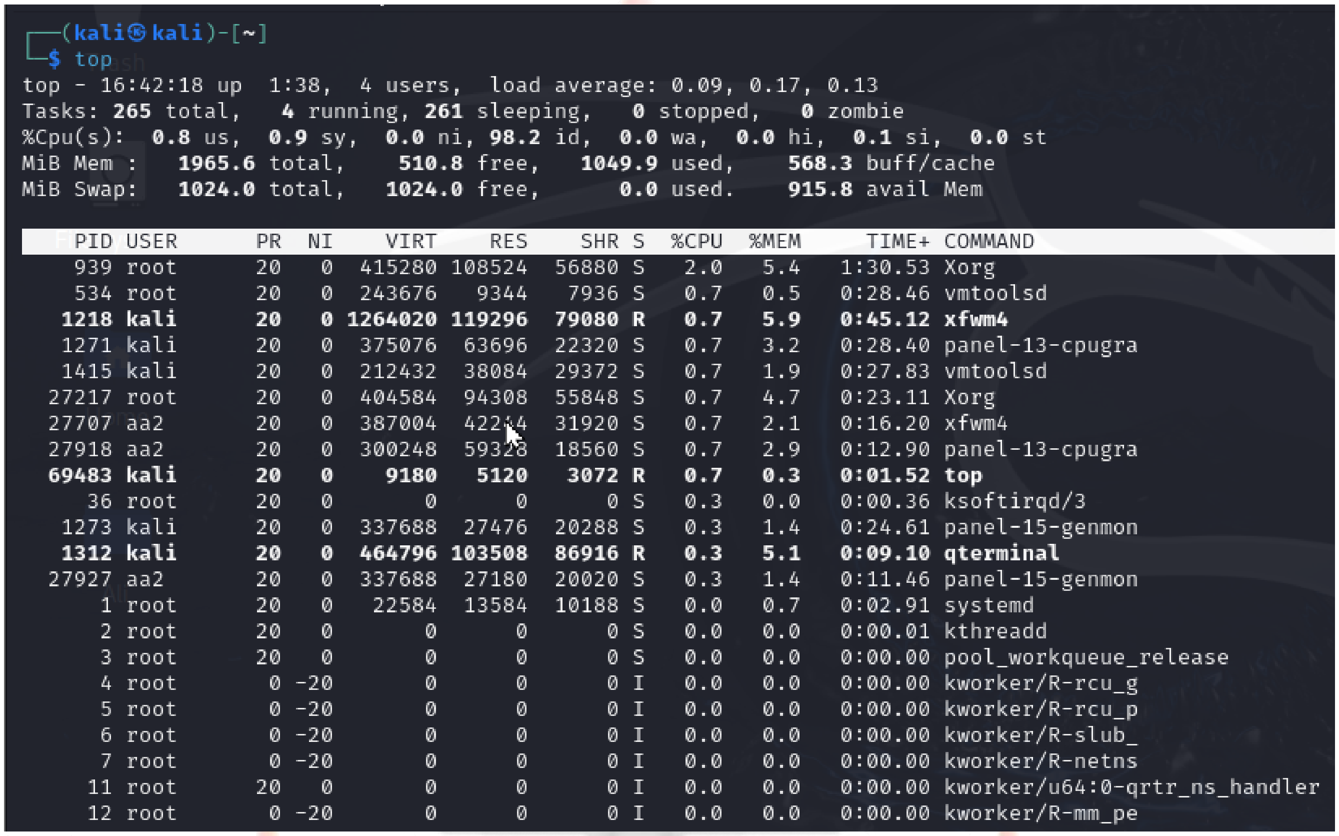
System Maintenance:

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
```

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.

These commands are essential for managing and controlling services on a Linux system.

sudo systemctl restart service_name This command is used to restart the specified service.

sudo systemctl start service_name This command starts the service if it is not running.

sudo systemctl stop service_name This command stops the specified service.

sudo systemctl status service_name This command shows the status of the service to see if it is running or not, along with additional recent logs.

sudo systemctl enable service_name This command is used to activate the specified

To Grant a User 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).

1. **Add the user to the sudo group:**

   ```
   sudo usermod -aG sudo username
   ```

2. **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 the Username:

To change the login name (username):

```
sudo usermod -l new_username old_username
```

- The -l option is used to change the login name of the user.

To also change the user's home directory to match the new username:

```
sudo usermod -d /home/new_username -m new_username
```

- -d: This option specifies the path for the new home directory.
- -m: This option moves the contents of the old home directory to the new one.

Changing the Password:

To change the password of a user:

```
sudo passwd username
```

- This command allows you to change the password for a specified user.

Deleting a User:

```
sudo deluser username
```

- ❖ This command removes the specified user from the system.

To delete the user's home directory and all their files:

```
sudo deluser --remove-home username
```

- ❖ This command will remove both the user and their home directory along with all personal files.

Task Scheduling:

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.

To check if cron is activated:

```
sudo systemctl status cron
```

- ❖ If you see the message: 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:

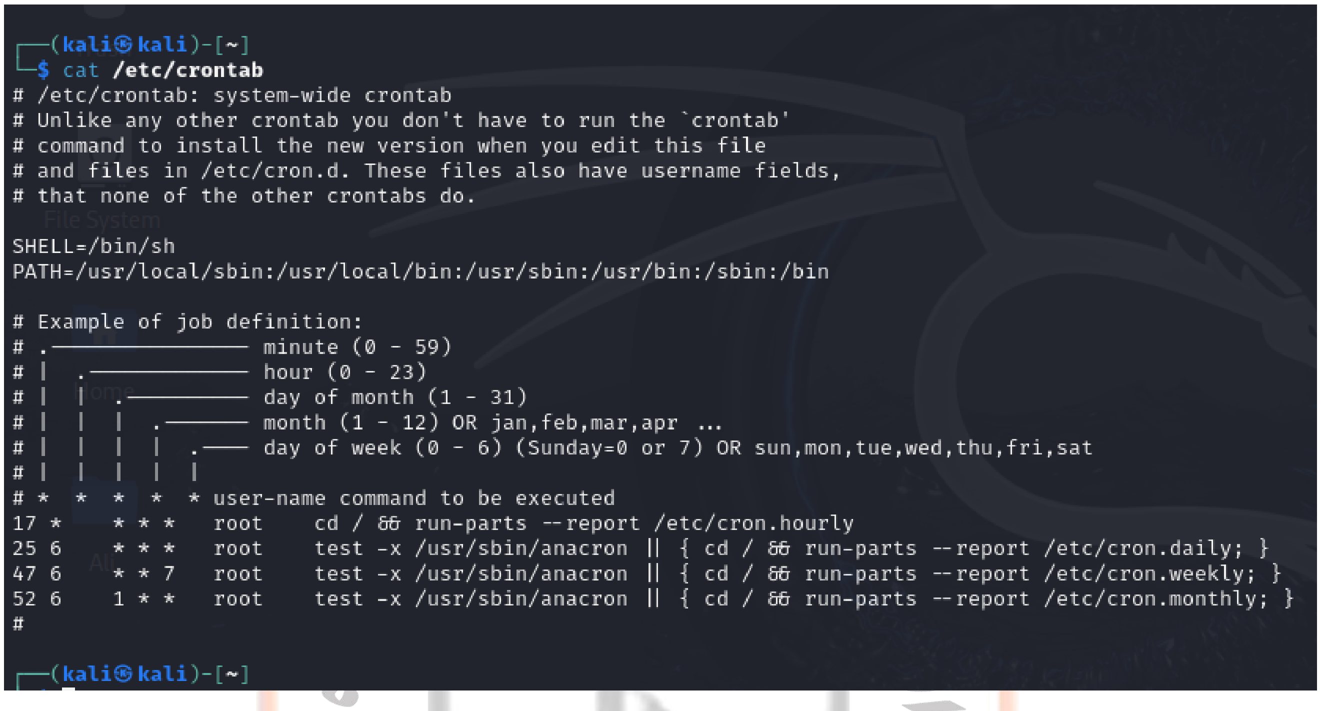
```
sudo systemctl enable cron
```

```
sudo systemctl start cron
```

Crontab (Task Scheduler Table):

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

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)-[~]
```

* * * * * **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:**
   - **crontab -e**
   - The -e option opens a text editor to input scheduled tasks.
2. **To display the scheduled tasks:**
   - **crontab -l**
   - The -l option lists all the scheduled tasks for the current user.
3. **To remove all scheduled tasks:**
   - **crontab -r**
   - The -r option removes all the scheduled tasks for the current user.

Scheduling Examples:

- **To create a folder on the desktop named BB1 after one minute:**

  ```
  * * * * mkdir /home/kali/Desktop/BB1
  ```
- To create a folder named BB1 after two minutes:

  ```
  */2 * * * * mkdir /home/kali/Desktop/BB1
  ```
- **To create a folder named BB1 after one and a half 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. **To run a task at a specific future time, such as creating a folder at 7:30 PM:**

   ```
   at 7:30 PM

   mkdir /home/kali/Desktop/BB12

   <press Ctrl+D>
   ```
3. **To view scheduled but not yet executed tasks:**

   atq   This command lists pending tasks in the queue.
4. **To remove a 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:**

at 10:00 AM Dec 15

Schedule a task to run after a specific duration (e.g., in 2 days):

at now + 2 days

Use other time formats, such as tomorrow or next week:

at 9:00 AM tomorrow

at 10:00 AM next week

Using at with echo to schedule via piping:

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

Software and Package Management in Linux:

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.

Updating repositories:

```
sudo apt update
```

Upgrading packages: To update all installed software to the latest versions:

```
sudo apt upgrade
```

Installing software: For example, to install the text editor Vim:

```
sudo apt install vim
```

Removing software:

```
sudo apt remove vim
```

This removes the program but keeps the personal configuration files.

Completely removing software (with configuration files):

```
sudo apt purge vim
```

Searching for packages:

```
apt search package_name
```

Example:

```
apt search nmap
```

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

```
sudo apt autoremove
```

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

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

Sometimes the system may have dependency issues (missing libraries or programs). If you encounter such a problem, you can fix it using the command:

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

Installing and upgrading a 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:

- ❖ **sudo apt install snapd**
- ❖ **sudo systemctl enable --now snapd**
- ❖ **sudo systemctl status snapd**
- ❖ **sudo systemctl restart snapd**

After that, you can install any program, for example:

- ❖ **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?**