

	Experiment No : 5
Title	To show various system configuration
Aim	Write a shell script to show various system configuration like currently logged user and his logname, your current shell, home directory, operating system type, current path setting, current working directory, show currently logged number of users, show memory information, Hard disk information like size of hard-disk, cache memory, model etc, and file system mounted
Hardware Requirement	Personal Computer
Software Requirement	Linux Operating System(Ubuntu 20.04) , Shell-Interpreter Nano or Vi or Vim or gedit text editor
Theory	<p>On Linux based system most of the hardware information can be extracted from /proc file system, for example display CPU and memory hardware information, enter the following cat command:</p> <pre>cat /proc/meminfo cat /proc/cpuinfo cat /proc/ide/had</pre> <p>Directories for System Information</p> <p>1./etc – Configuration Files</p> <ul style="list-style-type: none"> ▪ Contains configuration files required by all programs. ▪ This also contains startup and shutdown shell scripts used to start/stop individual programs. ▪ For example: /etc/resolv.conf, /etc/logrotate.conf <pre>cat /etc/sysconfig cat /etc/shell</pre> <p>1. /dev – Device Files</p> <ul style="list-style-type: none"> ▪ Contains device files. ▪ These include terminal devices, usb, or any device attached to the system. ▪ For example: /dev/tty1, /dev/usbmon0

2. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

3. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

Environmental and Shell Variables

- Each shell session keeps track of its own shell and environmental variables.
- Environment variables are name-value pairs used by a system's shell to define default shell properties.
- Some of these include your shell's home directory, prompt, and current working directory.
- Environment variables are inherited by sub-shells, and are available to applications, and daemons.
- One can create and set your own environment variables.
- These variables can access these in a few different ways.
- List of all of our environmental variables by using
- the `env` or `printenv` commands.
- In their default state, they should function exactly the same

There are some common environmental variables

- `SHELL`: This describes the shell that will be interpreting any commands you type in. In most cases, this will be bash by default, but other values can be set if you prefer other options.
- `TERM`: This specifies the type of terminal to emulate when running the shell. Different hardware terminals can be emulated for different operating requirements. You usually won't need to worry about this though.
- `USER`: The current logged in user.
- `PWD`: The current working directory.
- `OLDPWD`: The previous working directory. This is kept by the shell in order to switch back to your previous directory by running `cd -`.

- **LS_COLORS**: This defines color codes that are used to optionally add colored output to the **ls** command. This is used to distinguish different file types and provide more info to the user at a glance.
- **MAIL**: The path to the current user's mailbox.
- **PATH**: A list of directories that the system will check when looking for commands. When a user types in a command, the system will check directories in this order for the executable.
- **LANG**: The current language and localization settings, including character encoding.
- **HOME**: The current user's home directory.
- **_**: The most recent previously executed command.

In addition to these environmental variables, some shell variables that you'll often see are:

- **BASHOPTS**: The list of options that were used when bash was executed. This can be useful for finding out if the shell environment will operate in the way you want it to.
- **BASH_VERSION**: The version of bash being executed, in human-readable form.
- **BASH_VERSINFO**: The version of bash, in machine-readable output.
- **COLUMNS**: The number of columns wide that are being used to draw output on the screen.
- **DIRSTACK**: The stack of directories that are available with the **pushd** and **popd** commands.
- **HISTFILESIZE**: Number of lines of command history stored to a file.
- **HISTSIZE**: Number of lines of command history allowed in memory.
- **HOSTNAME**: The hostname of the computer at this time.
- **IFS**: The internal field separator to separate input on the command line. By default, this is a space.
- **PS1**: The primary command prompt definition. This is used to define what your prompt looks like when you start a shell session. The **PS2** is used to declare secondary prompts for when a command spans multiple lines.
- **SHELLOPTS**: Shell options that can be set with the **set** option.
- **UID**: The UID of the current user.

Displaying Environment Variables

To display the value of an environment variable, the **echo** command is employed, as demonstrated below:

Syntax:

echo \$VARIABLE_NAME

To obtain a list of all global environment variables, Linux offers several commands:

1. `printenv` Command in Linux

This command provides a comprehensive list of all global environment variables.

`printenv` //displays all the global ENVs

2. `set` Command in Linux

Lists all environment variables, encompassing both global and local variables.

`set` //display all the ENVs(global as well as local)

3. `env` Command in Linux

Presents a list of global environment variables.

`env` //display all the global ENVs

Differences Between Environment and Shell Variables

Standard UNIX variables are classified into two categories—

- environment variables and
- shell variables.

Environment variables:

- An environment variable is available and valid system-wide.
- Environment variables can be used by scripts and applications.
- These variables are inherited by all spawned child processes and shells.
- By convention, environment variables are given upper case names.

Shell variables:

- Shell variables are available only in the current shell session.
- These variables are useful when you need to store values temporarily.
- Each shell such as `zsh` and `bash`, has its own set of internal shell variables

Lshw(lost hardware)

Lshw is the abbreviation for **lost hardware**. The Lshw command is a command-line-based utility in Linux-based operating systems providing detailed information on the system's hardware configuration from various files in the `/proc` directory in your system.

The `lshw` command can also report the exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache memory configuration, bus speed, and many more.

The `lshw` command can print all the information mentioned above on a DMI-capable x86 or IA-64 (Itanium family of 64 microprocessors) system and some PowerPC machines. In a nutshell, the `lshw` is a small command that gives you a complete picture of your hardware configuration.

Syntax:

`lshw [-format] [-options ...]`

Where format can be:

- `-html`: Output hardware tree as HTML.
- `-xml`: Output hardware tree as XML.
- `-short`: Output hardware paths.
- `-businfo`: Output bus information.

Wc

wc (short for **word count**) is a command line tool in Unix/Linux operating systems, which is used to find out the number of newline count, word count, byte and character count in the files specified by the **File** arguments to the standard output and hold a total count for all named files.

When you define the **File** parameter, the **wc** command prints the file names as well as the requested counts. If you do not define a file name for the **File** parameter, it prints only the total count to the standard output.

wc Command Syntax

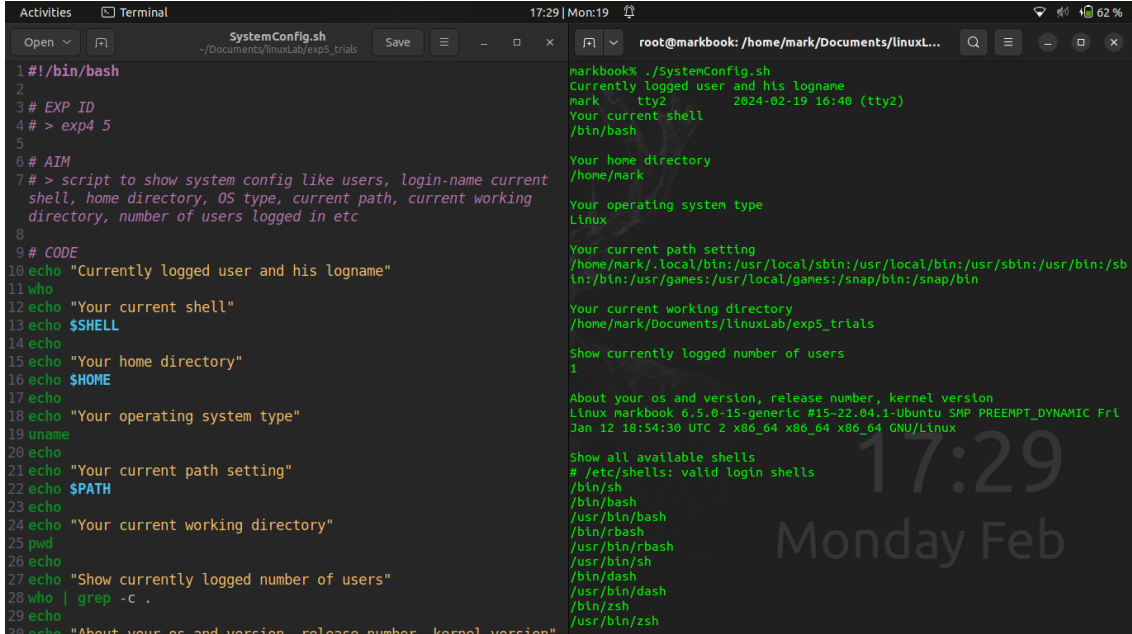
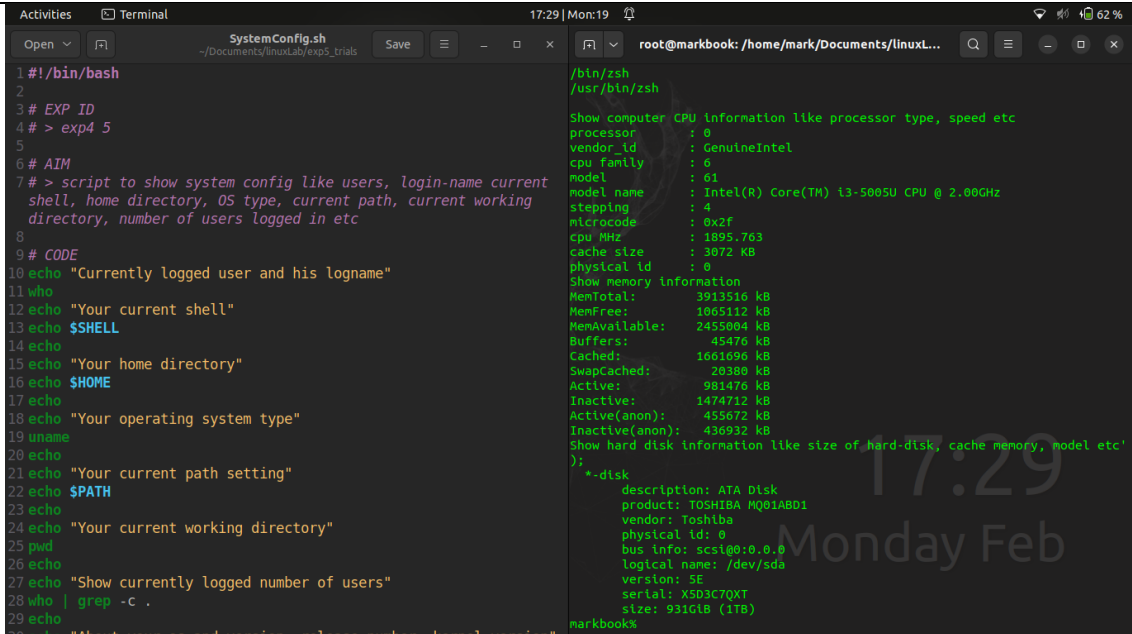
The syntax of the **wc** command is shown below.

```
# wc [options] filenames
```

The followings are the options and usage provided by the **wc** command.

- `wc -l` – Prints the number of lines in a file.
- `wc -w` – prints the number of words in a file.
- `wc -c` – Displays the count of bytes in a file.
- `wc -m` – prints the count of characters from a file.
- `wc -L` – prints only the length of the longest line in a file.

Shell Script	<pre>#!/bin/bash # EXP ID # > exp4 5 # AIM # > script to show system config like users, login-name current shell, home directory, OS type, current path, current working directory, number of users logged in etc # CODE echo "Currently logged user and his logname" who echo "Your current shell" echo \$SHELL echo echo "Your home directory" echo \$HOME echo echo "Your operating system type" uname echo echo "Your current path setting" echo \$PATH echo echo "Your current working directory" pwd echo echo "Show currently logged number of users" who grep -c . echo echo "About your os and version, release number, kernel version" uname -a echo echo "Show all available shells" cat /etc/shells echo # echo "Show mouse settings" # xev # this is commented out because it opens a new window and the script stops echo "Show computer CPU information like processor type, speed etc" cat /proc/cpuinfo head -n 10 echo "Show memory information"</pre>
---------------------	--

	<pre>cat /proc/meminfo head -n 10 echo "Show hard disk information like size of hard-disk, cache memory, model etc');"</pre> <pre>sudo lsblk -l head -n 10</pre>
Screenshot	 <pre> 1 #!/bin/bash 2 3 # EXP ID 4 # > exp4 5 5 6 # AIM 7 # > script to show system config like users, login-name current shell, home directory, OS type, current path, current working directory, number of users logged in etc 8 9 # CODE 10 echo "Currently logged user and his logname" 11 who 12 echo "Your current shell" 13 echo \$SHELL 14 echo 15 echo "Your home directory" 16 echo \$HOME 17 echo 18 echo "Your operating system type" 19 uname 20 echo 21 echo "Your current path setting" 22 echo \$PATH 23 echo 24 echo "Your current working directory" 25 pwd 26 echo 27 echo "Show currently logged number of users" 28 who grep -c . 29 echo 30 echo "About your os and version, release number, kernel version"</pre> <pre> markbook% ./SystemConfig.sh Currently logged user and his logname mark ttyz 2024-02-19 16:40 (tty2) Your current shell /bin/bash Your home directory /home/mark Your operating system type Linux Your current path setting /home/mark/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sb ln:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin Your current working directory /home/mark/Documents/linuxLab/exp5_trials Show currently logged number of users 1 About your os and version, release number, kernel version Linux markbook 6.5.0-15-generic #15-22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Jan 12 18:54:38 UTC 2 x86_64 x86_64 x86_64 GNU/Linux Show all available shells # /etc/shells: valid login shells /bin/sh /bin/bash /usr/bin/bash /bin/rbash /usr/bin/rbash /usr/bin/sh /bin/dash /usr/bin/dash /bin/zsh /usr/bin/zsh </pre>
	 <pre> /bin/zsh /usr/bin/zsh Show computer CPU information like processor type, speed etc processor : 0 vendor_id : GenuineIntel cpu family : 6 model : 61 model name : Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz stepping : 4 microcode : 0x2f cpu MHz : 1895.763 cache size : 3072 KB physical id : 0 Show memory information MemTotal: 3913516 kB MemFree: 1065112 kB MemAvailable: 2455004 kB Buffers: 45476 kB Cached: 1661096 kB SwapCached: 20380 kB Active: 981476 kB Inactive: 1474712 kB Active(anon): 455672 kB Inactive(anon): 436932 kB Show hard disk information like size of hard-disk, cache memory, model etc')) *-disk description: ATA Disk product: TOSHIBA MQ01ABD1 vendor: Toshiba physical id: 0 bus info: scsi@0:0.0 logical name: /dev/sda version: SE serial: X5D3C7QXT size: 931GiB (1TB) markbook% </pre>
Conclusion	<p>Learned to study system properties and configurations and process information in Linux. Also understood about environment variables.</p>

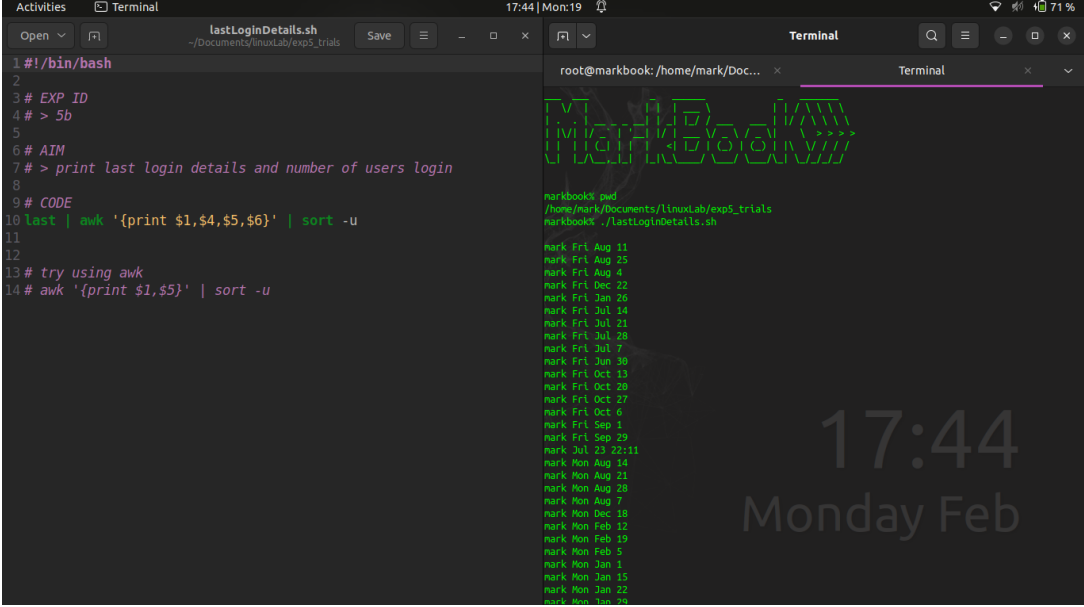
Signature	
Date	
Grade	

	Experiment No : 5A	Date :
Title	AddUser and Password	
Aim	Write a shell script to add user and password on Linux system.	
Hardware Requirement	Personal Computer	
Software Requirement	Linux Operating System(Ubuntu 20.04) , Shell-Interpreter Nano or Vi or Vim or gedit text editor	
Theory	<p>A system administrator manages configuration, upkeep and reliable operations of computer operations. Sysadmin handles servers, has to manage system performance and security without exceeding the budget to meet users need.</p> <p>A system administrator only deals with terminal interface and hence it is very important to learn and become master in commands to operate from terminal.</p> <p>Linux is a multi-user system, which means that more than one person can interact with the same system at the same time. As a system administrator, you have the responsibility to manage the system's users and groups by creating and removing users and assign them to different groups .</p> <p>In the /etc directory, the passwd and the group files hold all of the users and group information. These files are essential when logging on to the system. Anytime you add a user to a group in Linux, that user is added to the passwd file.</p> <p>The /etc/passwd files keep all the important information that is necessary for user login. The /etc/passwd file stores the user's account details.</p> <p>This file is a plain text file that contains a complete list of all users on your Linux system. It has the information about username, password, UID (user id), GID (group id), shell, and home directory.</p> <p>This file should have read permissions as many command-line utilities are used to map the user IDs to the user name. But, should have limited write access permissions only for superuser or root user accounts.</p>	

	<p>The /etc/group file holds all of the group information, as well as the users belonging to each group.</p>
Shell Script	<pre>#!/bin/bash # EXP ID # > exp4 5a # AIM # > Write a shell script to add user and password on Linux system # CODE # Am i Root user? if [\$(id -u) -eq 0]; then read -p "Enter username : " username read -s -p "Enter password : " password egrep "^\$username" /etc/passwd >/dev/null if [\$? -eq 0]; then # In a shell script, \$? is a special variable that holds the exit status of the last # command that was executed. echo "\$username exists!" exit 1 else pass=\$(perl -e 'print crypt(\$ARGV[0], "password")' \$password) useradd -m -p "\$pass" "\$username" [\$? -eq 0] && echo "User has been added to system!" echo "Failed to add a user!" fi else echo "Only root may add a user to the system." exit 2 fi # do something for password change feature # The command egrep "^\$username" /etc/passwd >/dev/null is used to check if a user # exists in a Unix-like system. Here's a breakdown of what it does: # egrep is a command-line utility for searching files for lines that match a pattern and # then printing the results. It's similar to grep, but supports extended regular # expressions. # "^\$username" is the pattern that egrep is searching for. The ^ symbol means the # start of a line. So, egrep is looking for any line in the file that starts with the value of # the variable username.</pre>

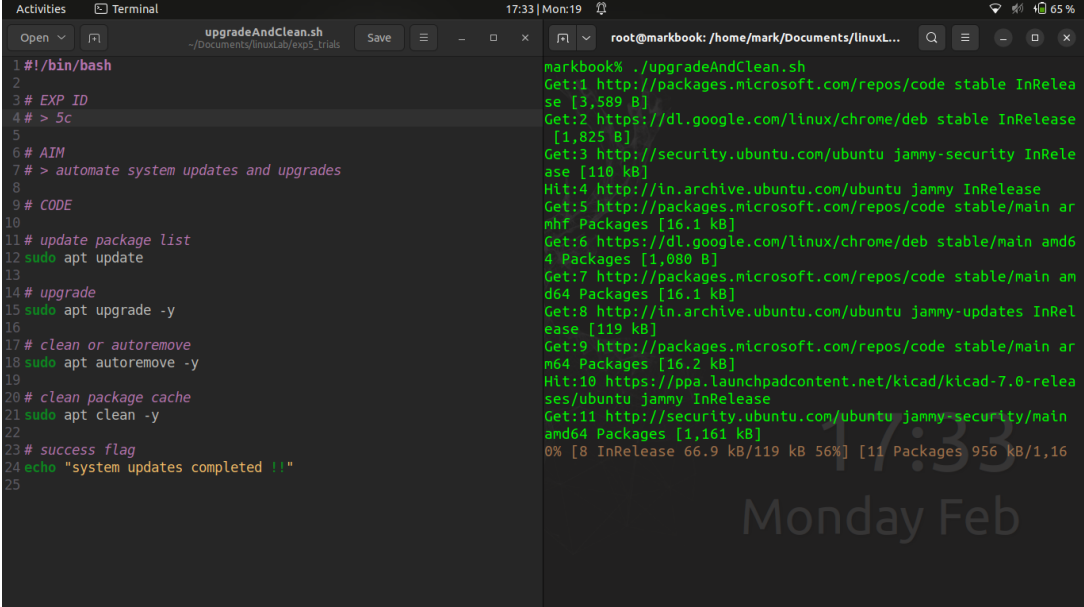
Signature	
Grade	
Date	

	Experiment No : 5B	Date :
Title	Last Login Details	
Aim	Write a shell script to print last login details	
Hardware Requirement	Personal Computer	
Software Requirement	Linux Operating System(Ubuntu 20.04) , Shell-Interpreter Nano or Vi or Vim or gedit text editor	
Theory	<p>The /var/log/lastlog file stores user last login information. This is binary file and act as database times of previous user logins. You need to use lastlog command to formats and prints the contents of the last login log /var/log/lastlog</p> <p>The easiest way to find the last login on your Linux computer is to execute the "last" command with no options. Using this command, last logins details performed on the computer can be found.</p>	
Shell Script	<pre>#!/bin/bash # EXP ID # > 5b # AIM # > print last login details and number of users login # CODE last awk '{print \$1,\$4,\$5,\$6}' sort -u # try using awk # awk '{print \$1,\$5}' sort -u</pre>	

Output	 <pre> 1 #!/bin/bash 2 3 # EXP ID 4 # > 5b 5 6 # AIM 7 # > print last login details and number of users login 8 9 # CODE 10 last awk '{print \$1,\$4,\$5,\$6}' sort -u 11 12 13 # try using awk 14 # awk '{print \$1,\$5}' sort -u </pre> <pre> markbook\$ pwd /home/mark/Documents/linuxlab/exp5_trials markbook\$./lastLoginDetails.sh mark Fri Aug 11 mark Fri Aug 25 mark Fri Aug 4 mark Fri Dec 22 mark Fri Jan 26 mark Fri Jul 14 mark Fri Jul 21 mark Fri Jul 28 mark Fri Jul 7 mark Fri Jun 30 mark Fri Oct 13 mark Fri Oct 20 mark Fri Oct 27 mark Fri Oct 6 mark Fri Sep 1 mark Fri Sep 29 mark Jul 23 22:11 mark Mon Aug 14 mark Mon Aug 21 mark Mon Aug 28 mark Mon Aug 7 mark Mon Dec 18 mark Mon Feb 12 mark Mon Feb 19 mark Mon Feb 5 mark Mon Jan 1 mark Mon Jan 15 mark Mon Jan 22 mark Mon Jan 29 </pre>
Conclusion	Learned about login log file /var/log/laslog and the last command.
Signature	
Grade	
Date	
	Experiment No : 5C
	Date:
Title	Upgrade and cleans the system automatically
Aim	Write a shell script to upgrade and cleans the system automatically instead of doing it manually.
Hardware Requirement	Personal Computer

Software Requirement	Linux Operating System(Ubuntu 20.04) , Shell-Interpreter Nano or Vi or Vim or gedit text editor
Theory	<p>As a regular user or system administrator, the package management tools apt or apt-get can be used in Linux. These package management tools can be used in order to manage certain operations such as searching for available packages, installing new packages, removing the existing ones, updating, and upgrading the installed packages, etc.</p> <p>Updating the packages : Linux operating systems come with a lot of free software updates for each package. It continuously releases the updates, patches, and fixes in order to improve the performance and fix bugs in them. It is very important to regularly check for these updates and install them in order to safeguard the system against potential threats and vulnerabilities. For installation of these updates, upgrade is performed and there are two ways to achieve this:</p> <ul style="list-style-type: none"> • apt-get upgrade and • apt-get dist-upgrade. <p>Upgrading Package Database</p> <p>To keep your system up to date, update and upgrade commands are used. The update command only updates the package list with the latest available versions, however, it does not install or upgrade the package. The upgrade command actually upgrades and installs the latest versions of packages that are already installed. Before going to upgrade the packages, check for the updates as follows. It will let the apt-get to know the new versions available.</p> <p>\$ sudo apt-get update</p> <p>Apt-get upgrade</p> <p>To install the latest versions of all the previously installed packages on your system, apt-get upgrade is used. This command only upgrades the packages which have a new release available as stated in the sources.list file in the "/etc/apt" folder. It does not attempt to install a new package or remove any installed package on its own.</p> <p>To upgrade or install the latest versions, run the following command as sudo as an only privilege user can check for and install updates on the Linux system:</p> <p>\$ sudo apt-get upgrade</p>

	<p>Apt-get dist-upgrade</p> <p>Similar to apt-get upgrade command, the apt-get dist-upgrade also upgrades the packages. In addition to this, it also handles changing dependencies with the latest versions of the package. It intelligently resolves the conflict among package dependencies and tries to upgrade the most significant packages at the expense of less significant ones, if required. Unlike apt-get upgrade command, the apt-get dist-upgrade is proactive and it installs new packages or removes existing ones on its own in order to complete the upgrade.</p> <p>In order to upgrade the packages, run the dist-upgrade command with sudo privileges:</p> <pre>\$ sudo apt-get dist-upgrade</pre> <p>To upgrade a specific package, command is as follows:</p> <pre>\$ sudo apt-get dist-upgrade <package_name></pre> <p>Sometimes, when you run apt-get upgrade, you receive the message "The following packages have been kept back". These packages are kept back because in order to install their new version, they need some other package which is not already installed. The apt-get upgrade only upgrades the existing packages, neither installs a new package or removes an existing one. That is why it keeps these packages back. Sometimes, the packages are also kept back because of broken dependencies (when the package which it depends on does not have a downloadable version)</p>
Shell Script	<pre>#!/bin/bash # EXP ID # > 5c # AIM # > automate system updates and upgrades # CODE # update package list sudo apt update # upgrade sudo apt upgrade -y</pre>

	<pre># clean or autoremove sudo apt autoremove -y # clean package cache sudo apt clean -y # success flag echo "system updates completed !!"</pre>
Output	

Grade	
Date	