Ex. No.: 2a EMBEDDED C CODE TO BLINKING LEDS

AIM:

To design and compile the Embedded C source code to control LEDs using Keil uVision4 compiler.

APPARATUS REQUIRED:

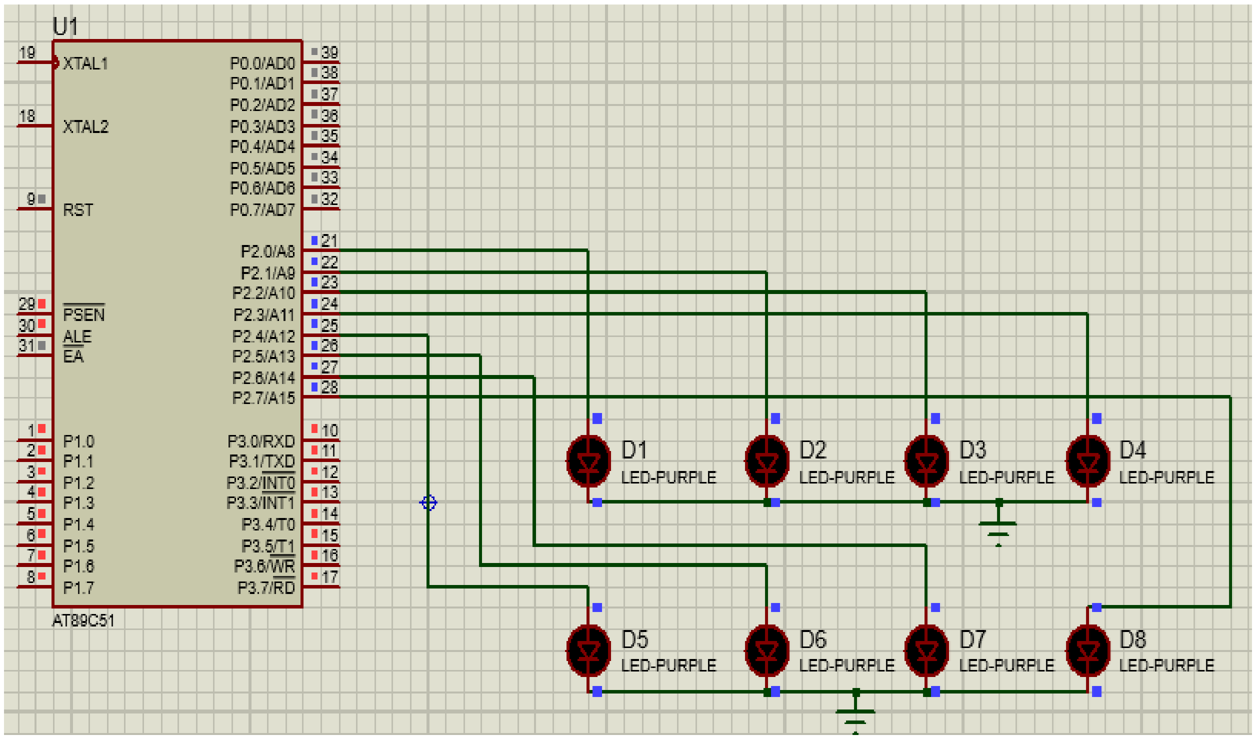
Keil uVision4 compiler and Proteus.

DEFINITIONS /THEORY:

Light Emitting Diodes (LEDs) are simple and most commonly used electronic components to display the digital signal states. The LED emits light when current is passed through it. It could blow up if we pass more current, hence we put a current limiting resistor. Usually, 220, 470 and 1K ohm resistors are commonly used as limiting resistors. You can use any of these depending on the required brightness. Lets, start blinking with LEDs and then generate the different patterns using the available LEDs .
The basic and important feature of any controllers is the number of GPIO's available for connecting the peripherals. 8051 has 32-gpio's grouped into 4-Ports namely P0-P3 as shown in the below table.

PORT	Number of Pins	Alternative Function
P0	8 (P0.0-P0.7)	AD0-AD7 (Address and Data bus)
P1	8 (P1.0-P1.7)	None
P2	8 (P2.0-P2.7)	A8-A15 (Higher Address Bus)
P3	8 (P3.0-P3.7)	UART, Interrupts, (T0/T1)Counters

Circuit Diagram



Lab Work:

PROGRAM

```
#include<reg51.h>
void delay()
{
int t;
for(t=0;t<32000;t++);
}
void main()
{
while(1)
{
P2=0xff;
delay();
P2=0x00;
delay();
}
}
```

OUTPUT:

Inference:

Result

Thus the compilation of the Embedded C source code to control LEDs using 8051 is done using Keil uVision compiler.

Ex. No.: 3a KEYPAD AND DISPLAY INTERFACING MICROCONTROLLER

AIM:

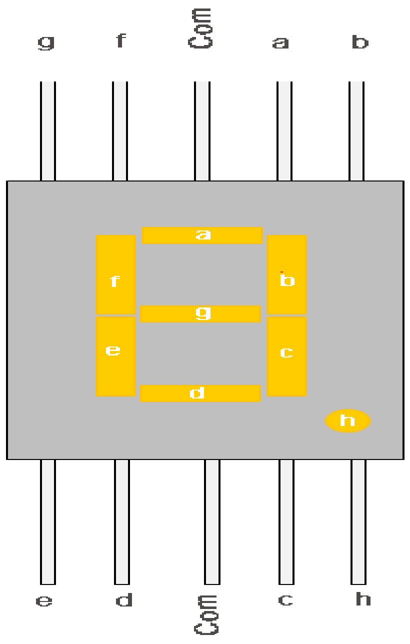
To design and compile the source code for any embedded system application using Keil uVision4 compiler.

APPARATUS REQUIRED:

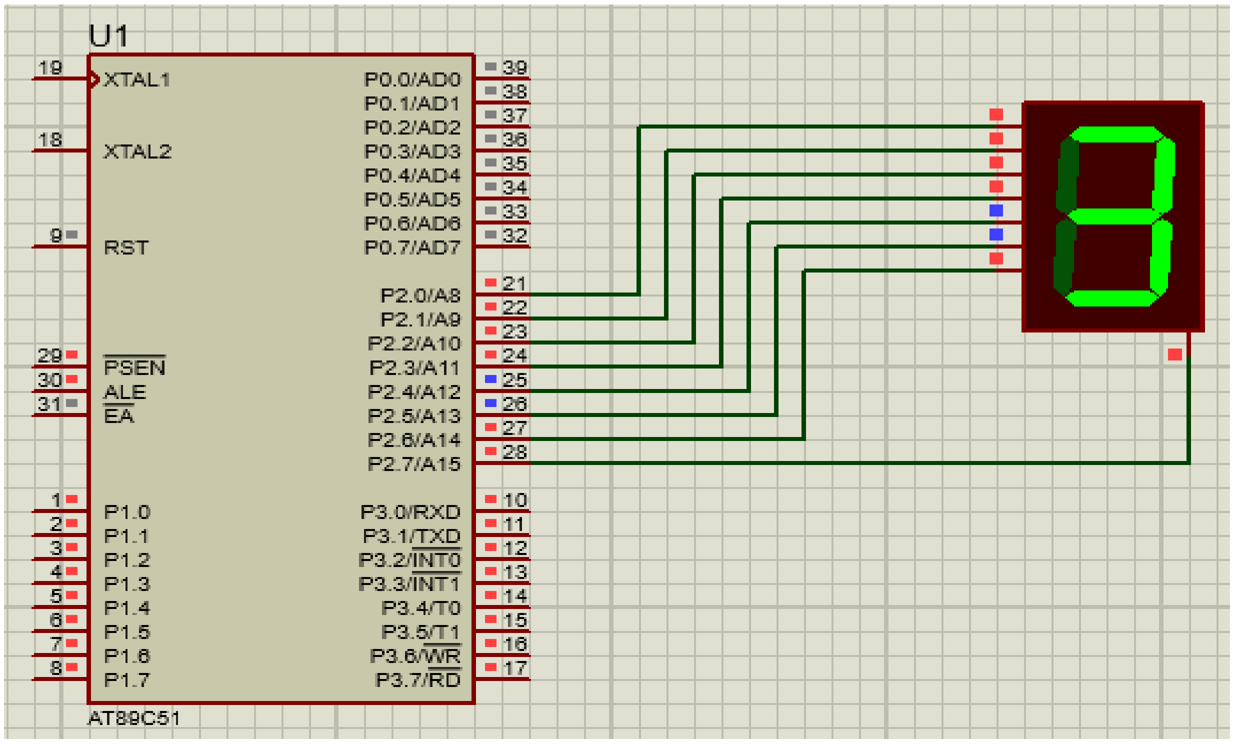
Keil uVision4 compiler and Proteus.

DEFINITIONS /THEORY:

Seven segment displays are used to indicate numerical information. Seven segments display can display digits from 0 to 9 and even we can display few characters like A, b, C, H, E, e, F, etc. These are very popular and have many more applications. So, in this project, I’ll show you how a 7 Segment Display works by interfacing 7 Segment Display to 8051 Microcontroller.



Circuit Diagram



Lab Work:

PROGRAM

```
ORG 00H
START:MOV R1,#10
MOV DPTR,#400H
BACK:CLR A
MOVC A,@A+DPTR
MOV P2,A
ACALL DELAY
INC DPTR
DJNZ R1,BACK
SJMP START
ORG 400H
DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
```

```
DELAY:MOV R2,#08H
UP2:MOV R4,#0FFH
UP1:MOV R3,#0FFH
HERE:DJNZ R3,HERE
DJNZ R4,UP1
DJNZ R2,UP2
RET
END
```

OUTPUT:

Inference:

Result

The design and compile the source code for any embedded system application of Seven Segment Display using 8051 is done using Keil uVision4 compiler.

Ex. No.: 3b **KEYPAD INTERFACING MICROCONTROLLER**

AIM:

To design and compile the source code for any embedded system application using Keil uVision4 compiler.

APPARATUS REQUIRED:

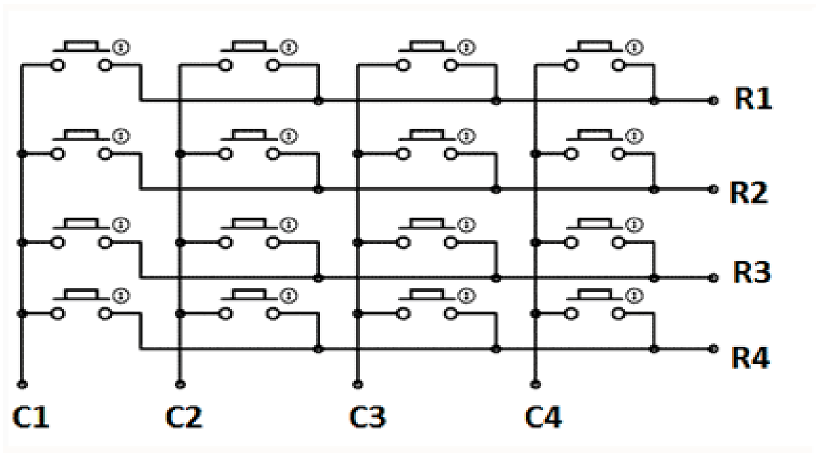
Keil uVision4 compiler and Proteus.

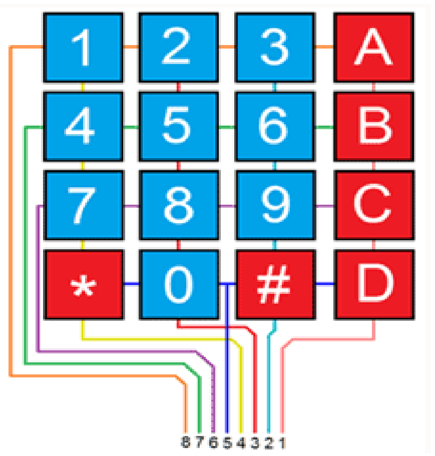
DEFINITIONS /THEORY:

Keypads are widely used input devices being used in various electronics and embedded projects. They are used to take inputs in the form of numbers and albhabets, and feed the same into system for further processing. In this tutorial we are going to interface a 4x4 matrix keypad with 8051 microcontroller**.**

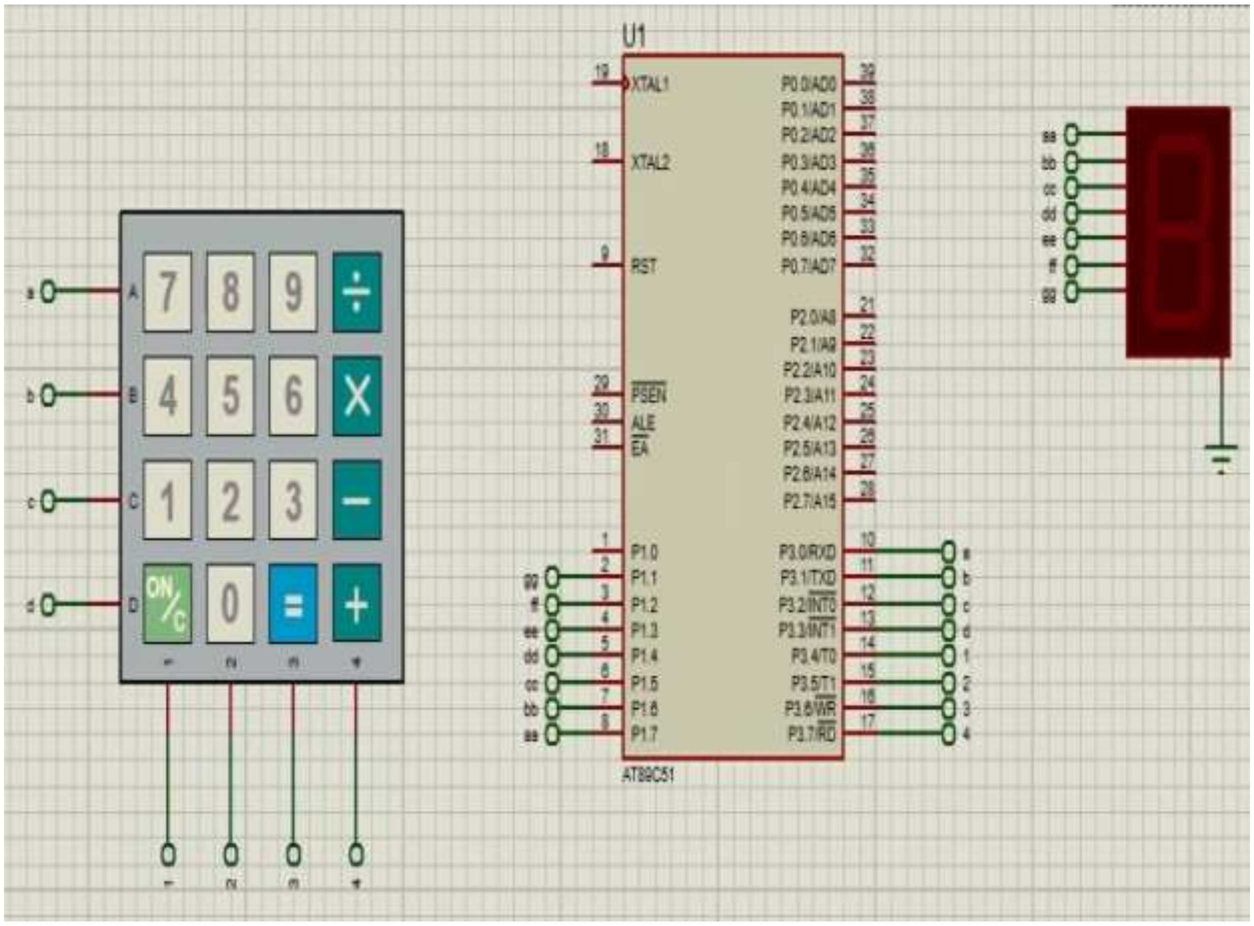
4X4 Matrix Keypad

Before we interface the keypad with microcontroller, first we need to understand how it works. Matrix keypad consists of set of Push buttons, which are interconnected. Like in our case we are using 4X4 matrix keypad, in which there are 4 push buttons in each of four rows. And the terminals of the push buttons are connected according to diagram. In first row, one terminal of all the 4 push buttons are connected together and another terminal of 4 push buttons are representing each of 4 columns, same goes for each row. So, we are getting 8 terminals to connect with a microcontroller.





Circuit Diagram



Lab Work:

PROGRAM

```
ORG 00H
MOV DPTR,#look_up_table
MOV A,#0FFH
MOV P1,#00000000B
reverse:MOV P3,#0FFH
CLR P3.0
JB P3.4,next_find_1
MOV A,#0D
ACALL disp_000
next_find_1:JB P3.5,next_find_2
MOV A,#1D
ACALL disp_000
next_find_2:JB P3.6,next_find_3
MOV A,#2D
ACALL disp_000
next_find_3:JB P3.7,next_find_4
MOV A,#3D
ACALL disp_000
next_find_4:SETB P3.0
CLR P3.1
JB P3.4,next_find_5
MOV A,#4D
ACALL disp_000
next_find_5:JB P3.5,next_find_6
MOV A,#5D
ACALL disp_000
next_find_6:JB P3.5,next_find_7
MOV A,#6D
ACALL disp_000
next_find_7:JB P3.5,next_find_8
MOV A,#7D
ACALL disp_000
next_find_8:SETB P3.1
CLR P3.2
JB P3.4,NEXT9
MOV A,#8D
ACALL disp_000
NEXT9:JB P3.5,next_find_10
MOV A,#9D
```

```
ACALL disp_000
next_find_10:JB P3.6,next_find_11
MOV A,#10D
ACALL disp_000
next_find_11:JB P3.7,next_find_12
MOV A,#11D
ACALL disp_000
next_find_12:SETB P3.2
CLR P3.3
JB P3.4, next_find_13
MOV A,#12D
ACALL disp_000
next_find_13:JB P3.5,next_find_14
MOV A,#13D
ACALL disp_000
next_find_14:JB P3.6,next_find_15
MOV A,#14D
ACALL disp_000
next_find_15:JB P3.7,reverse
MOV A,#15D
ACALL disp_000
LJMP reverse
disp_000:MOVC A,@A+DPTR
MOV P1,A

RET
look_up_table:
DB 11100000B
DB 11111110B
DB 11110110B
DB 10011100B
DB 01100110B
DB 10110110B
DB 10111110B
DB 00111110B
DB 01100000B
DB 11011010B
DB 11110010B
DB 11101110B
DB 10011110B
DB 11111100B
DB 10001110B
DB 01111010B
END
```

OUTPUT:

Inference:

Result

The design and compile the source code for any embedded system application of Interfacing Matrix Keyboard using 8051 is done using Keil uVision4 compiler.

Ex. No.: 5a STUDY BASIC AND USER STATUS LINUX COMMANDS

AIM:

To study the basics of linux commands.

DEFINITIONS /THEORY:

Inference:

Linux commands are a type of Unix command or shell procedure. They are the basic tools used to interact with Linux on an individual level. Linux commands are used to perform a variety of tasks, including displaying information about files and directories.

Linux operating system is used on servers, desktops, and maybe even your smartphone. It has a lot of command line tools that can be used for virtually everything on the system.

Linux Commands:

1. **ls** - The most frequently used command in Linux to list directories
2. **pwd** - Print working directory command in Linux
3. **cd** - Linux command to navigate through directories
4. **mkdir** - Command used to create directories in Linux
5. **mv** - Move or rename files in Linux
6. **cp** - Similar usage as mv but for copying files in Linux
7. **rm** - Delete files or directories
8. **touch** - Create blank/empty files
9. **ln** - Create symbolic links (shortcuts) to other files
10. **clear** - Clear the terminal display
11. **cat** - Display file contents on the terminal
12. **echo** - Print any text that follows the command
13. **less** - Linux command to display paged outputs in the terminal
14. **man** - Access manual pages for all Linux commands
15. **uname** - Linux command to get basic information about the OS
16. **whoami** - Get the active username
17. **tar** - Command to extract and compress files in linux
18. **grep** - Search for a string within an output
19. **head** - Return the specified number of lines from the top
20. **tail** - Return the specified number of lines from the bottom
21. **diff** - Find the difference between two files
22. **cmp** - Allows you to check if two files are identical
23. **comm** - Combines the functionality of diff and cmp
24. **sort** - Linux command to sort the content of a file while outputting
25. **export** - Export environment variables in Linux
26. **zip** - Zip files in Linux
27. **unzip** - Unzip files in Linux
28. **ssh** - Secure Shell command in Linux
29. **service** - Linux command to start and stop services

30. **ps** - Display active processes
31. **kill and killall** - Kill active processes by process ID or name
32. **df** - Display disk filesystem information
33. **mount** - Mount file systems in Linux
34. **chmod** - Command to change file permissions
35. **chown** - Command for granting ownership of files or folders
36. **ifconfig** - Display network interfaces and IP addresses
37. **traceroute** - Trace all the network hops to reach the destination
38. **wget** - Direct download files from the internet
39. **ufw** - Firewall command
40. **iptables** - Base firewall for all other firewall utilities to interface with
41. **apt, pacman, yum, rpm** - Package managers depending on the distribution
42. **sudo** - Command to escalate privileges in Linux
43. **cal** - View a command-line calendar
44. **alias -** Create custom shortcuts for your regularly used commands
45. **dd** - Majorly used for creating bootable USB sticks
46. **whereis** - Locate the binary, source, and manual pages for a command
47. **whatis** - Find what a command is used for
48. **top** - View active processes live with their system usage
49. **useradd and usermod** - Add a new user or change existing user data
50. **passwd** - Create or update passwords for existing users

1. pwd command

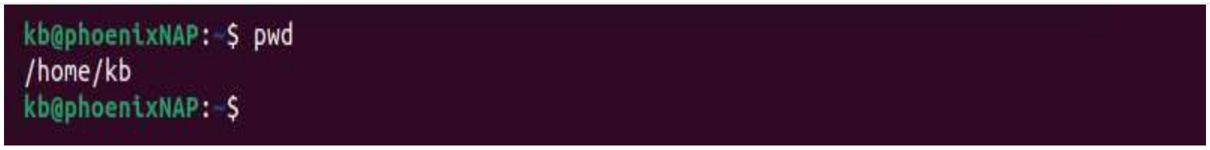
The pwd command (**p**rint **w**orking **d**irectory) is a shell builtin command that prints the current location. The output shows an absolute directory path, starting with the root directory (**/**).

The general syntax is:

```
pwd <options>
```

To see how the command works, run the following in the terminal:

```
pwd
```



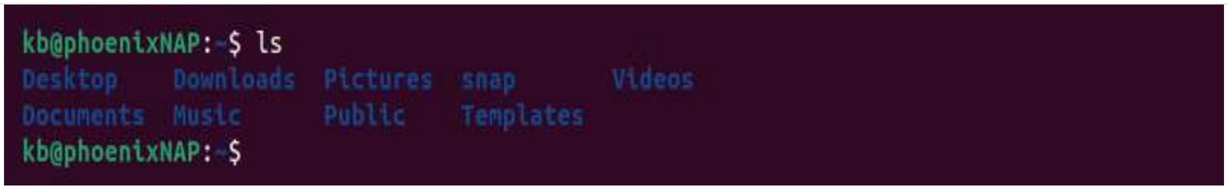
```
kb@phoenixNAP:~$ pwd
/home/kb
kb@phoenixNAP:~$
```

The output prints the current location in the **/home/<username>** format.

2. ls command

The ls command (**lis**t) prints a list of the current directory's contents. Run the following:

```
ls
```



```
kb@phoenixNAP:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
kb@phoenixNAP:~$
```

Additional options provide flexibility with the display output. Typical usage includes combining the following options:

- Show as a list:

```
ls -l
```

- Show as a list and include hidden files:

```
ls -la
```

- Show sizes in a human-readable format:

```
ls -lah
```

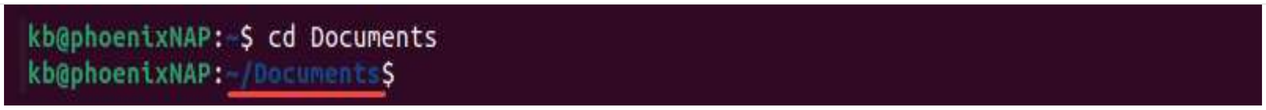
3. cd command

The cd command (change directory) is a shell builtin command for changing the current working directory:

```
cd <directory>
```

For example, to move to the *Document* directory, run:

```
cd Documents
```



```
kb@phoenixNAP:~$ cd Documents
kb@phoenixNAP:~/Documents$
```

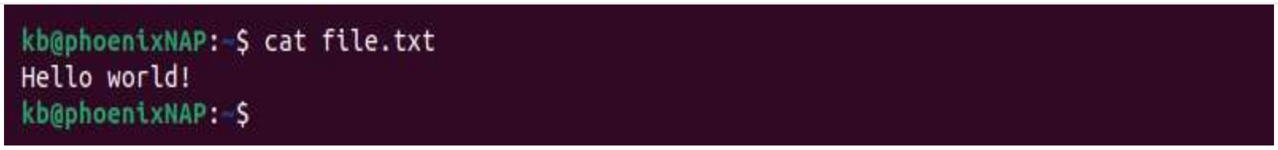
The working directory changes in the terminal interface. In a non-default interface, use the **pwd** command to check the current directory.

Use **cd** without any parameters to return to the home directory (**~**).

4. cat command

The cat command (con**cat**enate) displays the contents of a file in the terminal (standard output or stdout). To use the command, provide a file name from the current directory:

```
cat <filename>
```



```
kb@phoenixNAP:~$ cat file.txt
Hello world!
kb@phoenixNAP:~$
```

Alternatively, provide a path to the file along with the file name:

```
cat <path>/<filename>
```

The command can also:

- Display contents of multiple files:

```
cat <file 1> <file 2>
```

- Create new files:

```
cat ><filename>
```

Add contents to the file and press **CTRL**+**D** to exit.

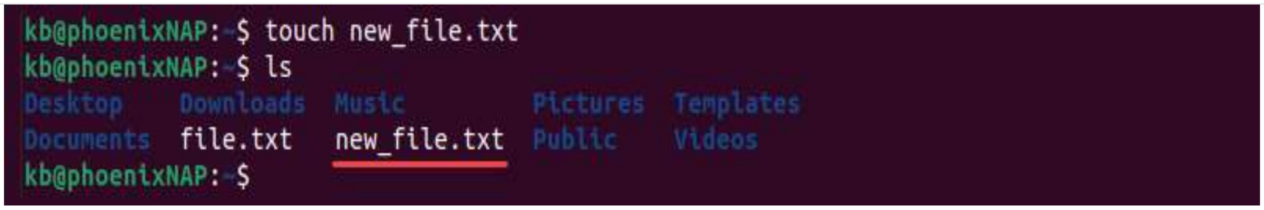
- Display line numbers:

```
cat -n <filename>
```

5. touch command

The primary purpose of the touch command is to modify an existing file's timestamp. To use the command, run:

```
touch <filename>
```



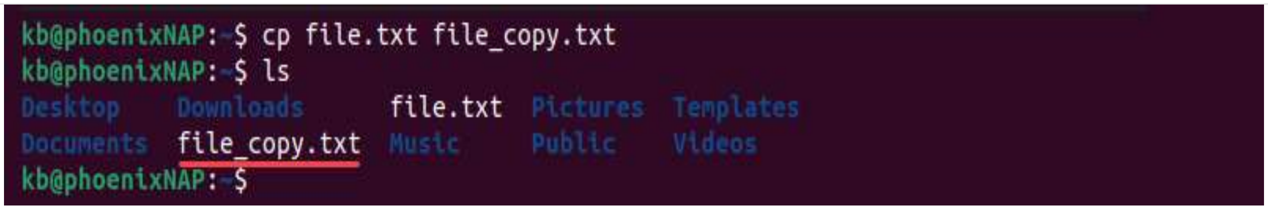
```
kb@phoenixNAP:~$ touch new_file.txt
kb@phoenixNAP:~$ ls
Desktop    Downloads  Music         Pictures  Templates
Documents  file.txt   new_file.txt  Public    Videos
kb@phoenixNAP:~$
```

The command creates an empty file if it does not exist. Due to this effect, **touch** is also a quick way to make a new file (or a batch of files).

6. cp command

The main way to copy files and directories in Linux is through the cp command (**cop**y). Try the command with:

```
cp <source file> <target file>
```



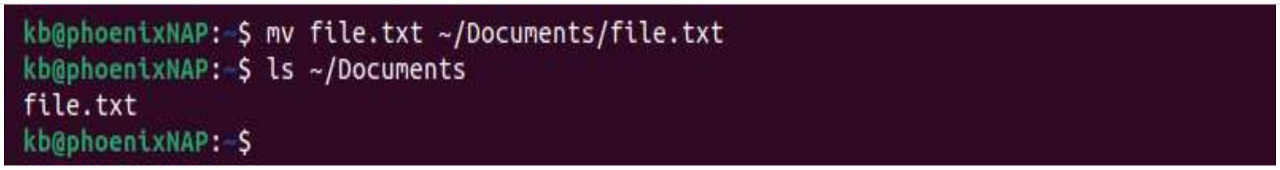
```
kb@phoenixNAP:~$ cp file.txt file_copy.txt
kb@phoenixNAP:~$ ls
Desktop    Downloads      file.txt  Pictures  Templates
Documents  file_copy.txt  Music     Public    Videos
kb@phoenixNAP:~$
```

The source and target files must have different names since the command copies in the same directory. Provide a path before the file name to copy to another location.

7. mv command

Use the mv command (**mov**e) to move files or directories from one location to another. For example, to move a file from the current directory to *~/Documents*, run:

```
mv <filename> ~/Documents/<filename>
```

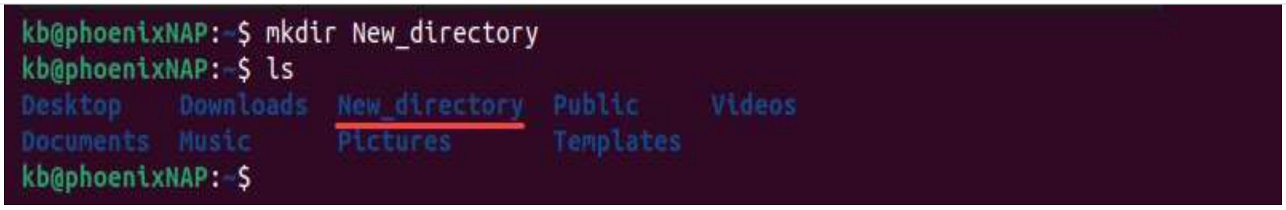


```
kb@phoenixNAP:~$ mv file.txt ~/Documents/file.txt
kb@phoenixNAP:~$ ls ~/Documents
file.txt
kb@phoenixNAP:~$
```

8. mkdir command

The mkdir command (**m**a**k**e **dir**ectory) creates a new directory in the provided location. Use the command in the following format:

```
mkdir <directory name>
```



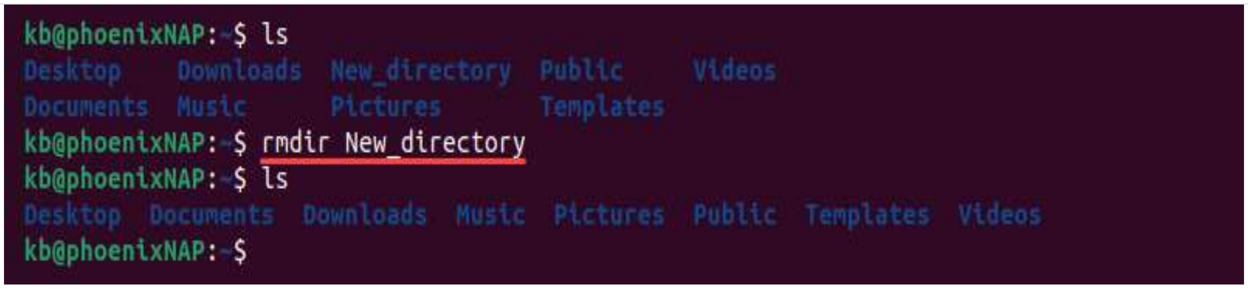
```
kb@phoenixNAP:~$ mkdir New_directory
kb@phoenixNAP:~$ ls
Desktop    Downloads  New_directory  Public     Videos
Documents  Music      Pictures       Templates
kb@phoenixNAP:~$
```

Provide a path to create a directory in the given location, or use a space or comma-separated list to create multiple directories simultaneously.

9. rmdir command

Use the rmdir command (**re**move **dir**ectory) to delete an empty directory. For example:

```
rmdir <directory name>
```



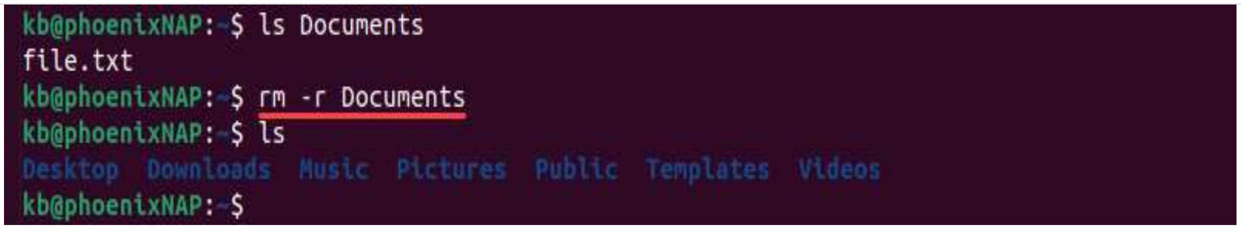
```
kb@phoenixNAP:~$ ls
Desktop    Downloads  New_directory  Public     Videos
Documents  Music      Pictures       Templates
kb@phoenixNAP:~$ rmdir New_directory
kb@phoenixNAP:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
kb@phoenixNAP:~$
```

If the directory is not empty, the command fails.

10. rm command

The **rm** command (**rem**ove) deletes files or directories. To use the command for non-empty directories, add the **-r** tag:

```
rm -r <file or directory>
```



```
kb@phoenixNAP:~$ ls Documents
file.txt
kb@phoenixNAP:~$ rm -r Documents
kb@phoenixNAP:~$ ls
Desktop  Downloads  Music  Pictures  Public  Templates  Videos
kb@phoenixNAP:~$
```

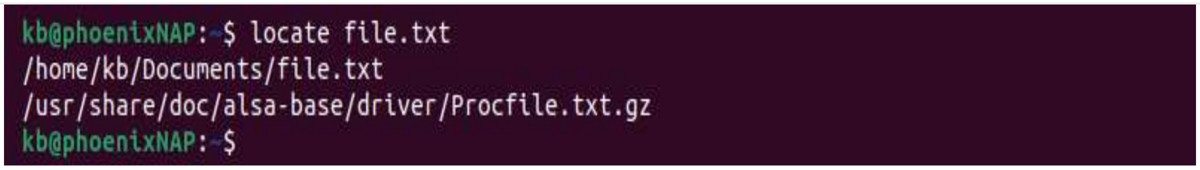
Unlike the **rmdir** command, **rm** also removes all the contents from the directory.

Note: Removing some directories in Linux is dangerous. Make sure you know what you're removing before running a dangerous Linux terminal command.
11. locate command

The locate command is a simple Linux tool for finding a file. The command checks a file database on a system to perform the search quickly. However, the result is sometimes inaccurate if the database is not updated.

To use the command, install locate and try the following example:

```
locate <filename>
```



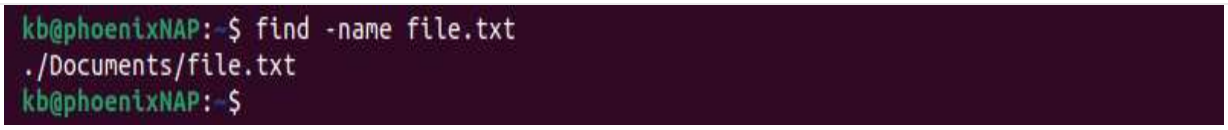
```
kb@phoenixNAP:~$ locate file.txt
/home/kb/Documents/file.txt
/usr/share/doc/alsa-base/driver/Procfile.txt.gz
kb@phoenixNAP:~$
```

The output prints the file's location path. The matching is unclear and outputs any file that contains the file name.

12. find command

Use the find command to perform a thorough search on the system. Add the **-name** tag to search for a file or directory by name:

```
find -name <file or directory>
```



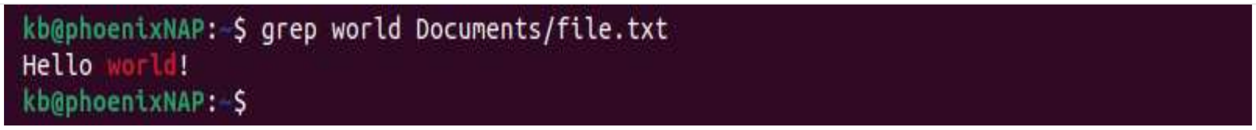
```
kb@phoenixNAP:~$ find -name file.txt
./Documents/file.txt
kb@phoenixNAP:~$
```

The output prints the file's path and performs an exact match. Use additional options to control the search further.

13. grep command

The grep command (**g**lobal **r**egular **e**xpression **p**rint) enables searching through text in a file or a standard output. The basic syntax is:

```
grep <search string> <filename>
```



```
kb@phoenixNAP:~$ grep world Documents/file.txt
Hello world!
kb@phoenixNAP:~$
```

The output highlights all matches. Advanced commands include using grep for multiple strings or writing grep regex statements.

14. sudo command

The sudo command (**s**uper**u**ser **do**) elevates a user's permissions to administrator or root. Commands that change system configuration require elevated privilege.

Add **sudo** as a prefix to any command that requires elevated privileges:

```
sudo <command>
```

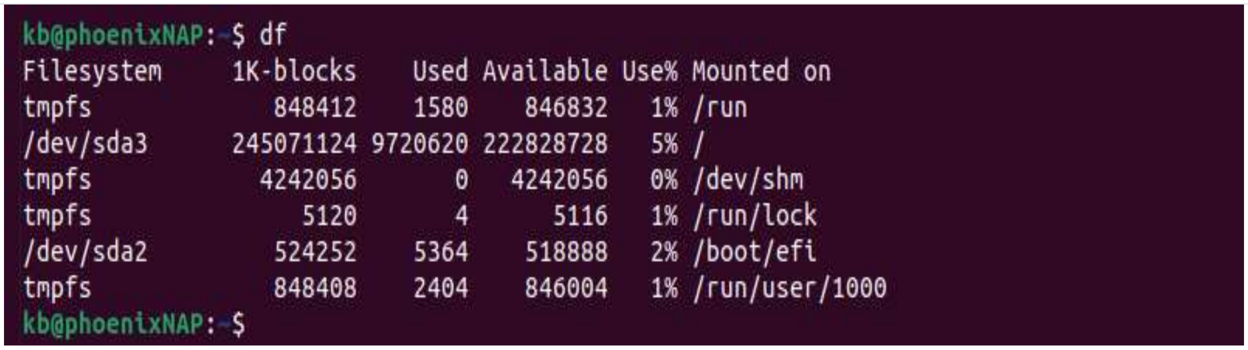
Use the command with caution to avoid making accidental changes permanent.

Note: Learn more about Linux file permissions.

15. df command

The **df** command (**d**isk **f**ree) is used to check available disk space on the file system. To see how **df** works, run the following:

```
df
```

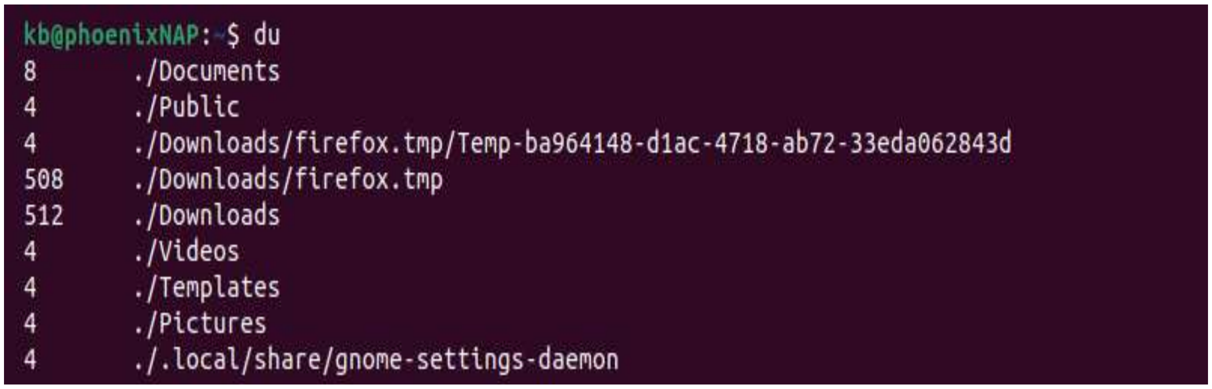


```
kb@phoenixNAP:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
tmpfs             848412    1580    846832   1% /run
/dev/sda3      245071124 9720620 222828728   5% /
tmpfs            4242056       0   4242056   0% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
/dev/sda2         524252    5364    518888   2% /boot/efi
tmpfs             848408    2404    846004   1% /run/user/1000
kb@phoenixNAP:~$
```

The output shows the amount of space used by different drives. Add the **-h** tag to make the output in human-readable format (kilobytes, megabytes, and gigabytes).
16. du command

The **du** (**d**isk **u**sage) command helps show how much space a file or directory takes up. Run the command without any parameters:

```
du
```



```
kb@phoenixNAP:~$ du
8       ./Documents
4       ./Public
4       ./Downloads/firefox.tmp/Temp-ba964148-d1ac-4718-ab72-33eda062843d
508     ./Downloads/firefox.tmp
512     ./Downloads
4       ./Videos
4       ./Templates
4       ./Pictures
4       ./.local/share/gnome-settings-daemon
```

The output shows the amount of space used by files and directories in the current directory. The size displays in blocks, and adding the **-h** tag changes the measure to human-readable format.

17. head command

Use the head command to truncate long outputs. The command can truncate files, for example:

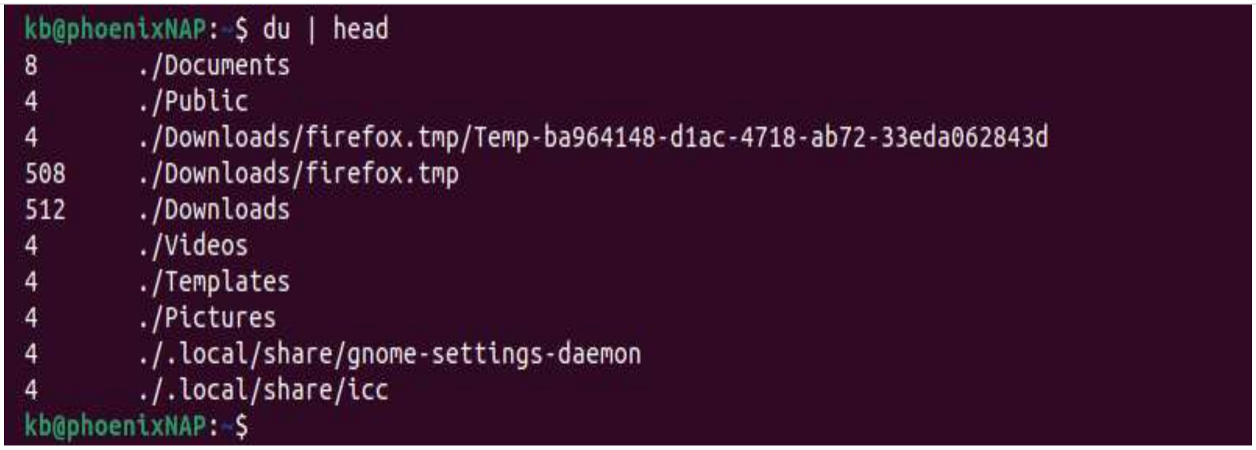
```
head <filename>
```

Alternatively, pipe **head** to a command with a long output:

```
<command> | head
```

For example, to see the first ten lines of the **du** command, run:

```
du | head
```



```
kb@phoenixNAP:~$ du | head
8       ./Documents
4       ./Public
4       ./Downloads/firefox.tmp/Temp-ba964148-d1ac-4718-ab72-33eda062843d
508     ./Downloads/firefox.tmp
512     ./Downloads
4       ./Videos
4       ./Templates
4       ./Pictures
4       ./.local/share/gnome-settings-daemon
4       ./.local/share/icc
kb@phoenixNAP:~$
```

The output shows the first ten lines instead of everything.

18. tail command

The Linux tail command does the opposite of **head**. Use the command to show the last ten lines of a file:

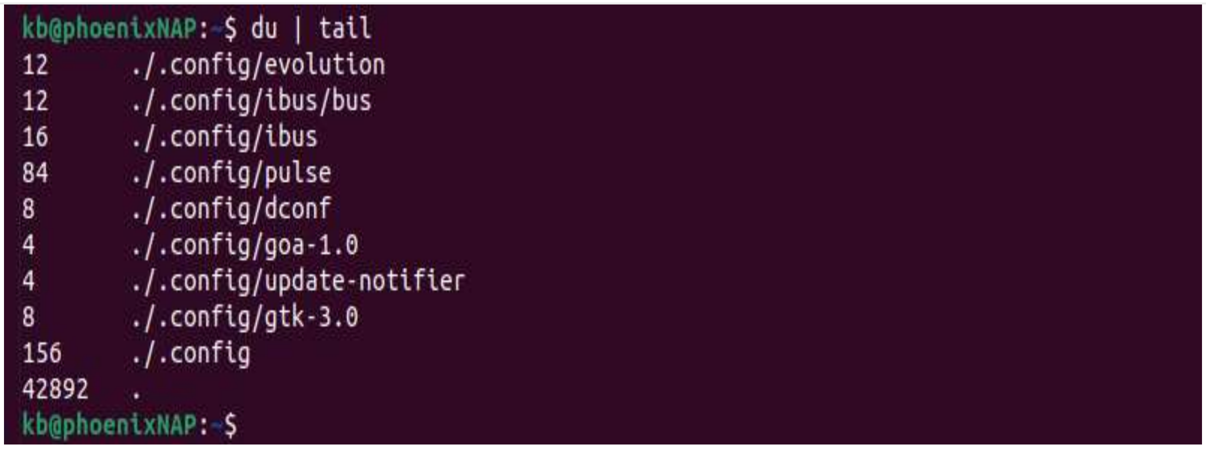
```
tail <filename>
```

Or pipe **tail** to a command with a long output:

```
<command> | tail
```

For example, use **tail** to see the last ten lines of the **du** command:

```
du | tail
```



```
kb@phoenixNAP:~$ du | tail
12      ./.config/evolution
12      ./.config/ibus/bus
16      ./.config/ibus
84      ./.config/pulse
8       ./.config/dconf
4       ./.config/goa-1.0
4       ./.config/update-notifier
8       ./.config/gtk-3.0
156     ./.config
42892   .
kb@phoenixNAP:~$
```

Both **head** and **tail** commands are helpful when reading Linux log files.

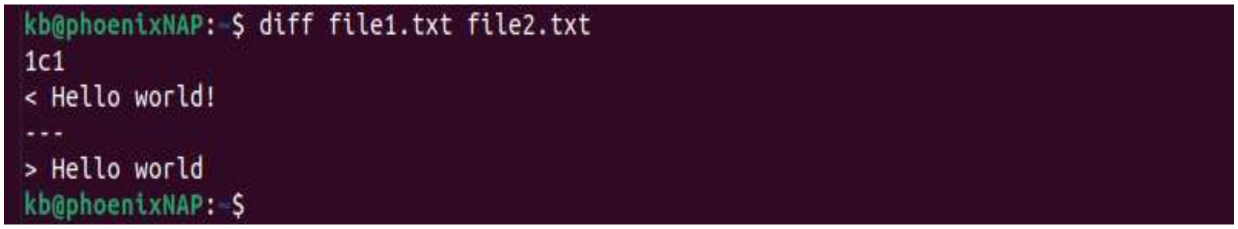
19. diff command

The diff command (**diff**erence) compares two files and prints the difference. To use the command, run:

```
diff <file 1> <file 2>
```

For example, to compare files *test1.txt* and *test2.txt*, run:

```
diff file1.txt file2.txt
```



```
kb@phoenixNAP:~$ diff file1.txt file2.txt
1c1
< Hello world!
---
> Hello world
kb@phoenixNAP:~$
```

Developers often use **diff** to compare versions of the same code.

Note: Learn how to utilize diff --color to change the color of the output.

20. tar command

The tar command (**t**ape **ar**chiver) helps archive, compress, and extract archived files.

The command manages and creates files known as **tarballs**, which often appear during installation processes. The options provide different functionalities depending on the task.
21. chmod command

Use the **chmod** (**ch**ange **mod**e) command to change file and directory permissions. The command requires setting the permission code and the file or directory to which the permissions apply.

For example:

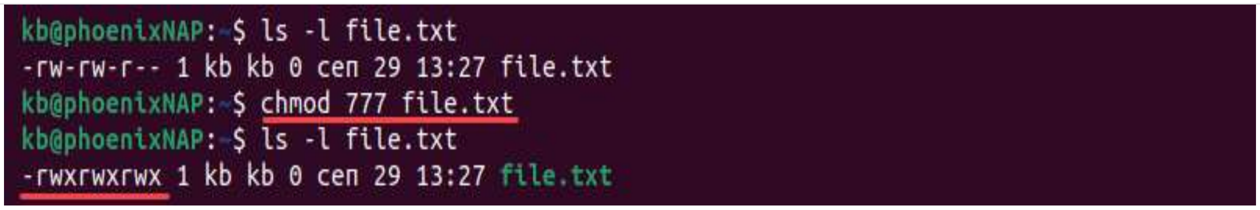
```
chmod <permission> <file or directory>
```

The permission is a number code consisting of three numbers:

- The first number is the permission of the current user (owner).
- The second number is the permission for the group.
- The third number is permissions for everyone else.

For example, to change the file permissions for a test.txt file so anyone can read, write, and execute, run:

```
chmod 777 file.txt
```



```
kb@phoenixNAP:~$ ls -l file.txt
-rw-rw-r-- 1 kb kb 0 сеп 29 13:27 file.txt
kb@phoenixNAP:~$ chmod 777 file.txt
kb@phoenixNAP:~$ ls -l file.txt
-rwxrwxrwx 1 kb kb 0 сеп 29 13:27 file.txt
```

Note: Allowing anyone to read, write, and execute files is considered a bad security practice. Implement privileged access management to maximize security on your system.

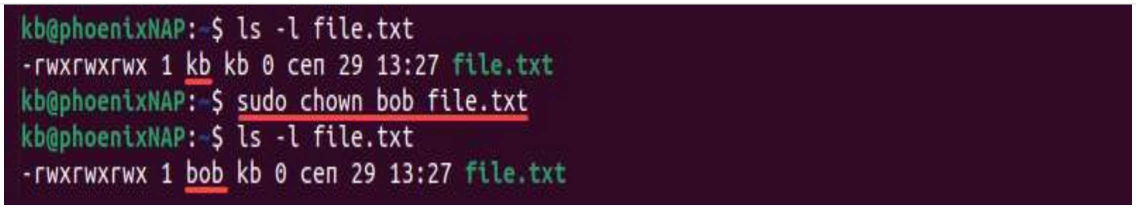
22. chown command

The chown command (**ch**ange **own**ership) changes the ownership of a file or directory. To transfer ownership, use the following command as sudo:

```
sudo chown <new owner name or UID> <file or directory>
```

For example:

```
sudo chown bob file.txt
```



```
kb@phoenixNAP:~$ ls -l file.txt
-rwxrwxrwx 1 kb kb 0 сеп 29 13:27 file.txt
kb@phoenixNAP:~$ sudo chown bob file.txt
kb@phoenixNAP:~$ ls -l file.txt
-rwxrwxrwx 1 bob kb 0 сеп 29 13:27 file.txt
```

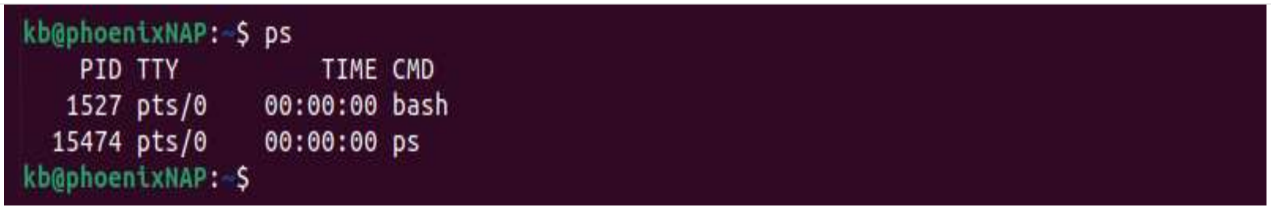
Configuring ownership is a common task during installations. The **chown** command allows daemons and processes to access files during setup.

23. ps command

The **ps** (process status) command lists processes currently running on the system. Every task creates a single or multiple processes running in the background.

Run **ps** without any options to see the running processes in the terminal session:

```
ps
```



```
kb@phoenixNAP:~$ ps
    PID TTY          TIME CMD
   1527 pts/0    00:00:00 bash
  15474 pts/0    00:00:00 ps
kb@phoenixNAP:~$
```

The output shows the process ID (PID), the terminal type, CPU time usage, and the command that started the process.

24. top command

The top command (**t**able **o**f **p**rocesses) is an extended version of the **ps** command. Run the command without any options to see the result:

```
top
```

The output lists all running processes in real-time. To exit the viewer, press **CTRL**+**C**.

25. kill command

Use the **kill** command to terminate an unresponsive process. The command syntax is:

```
kill <signal option> <process ID>
```

There are sixty-four different signal numbers, but the most commonly used are:

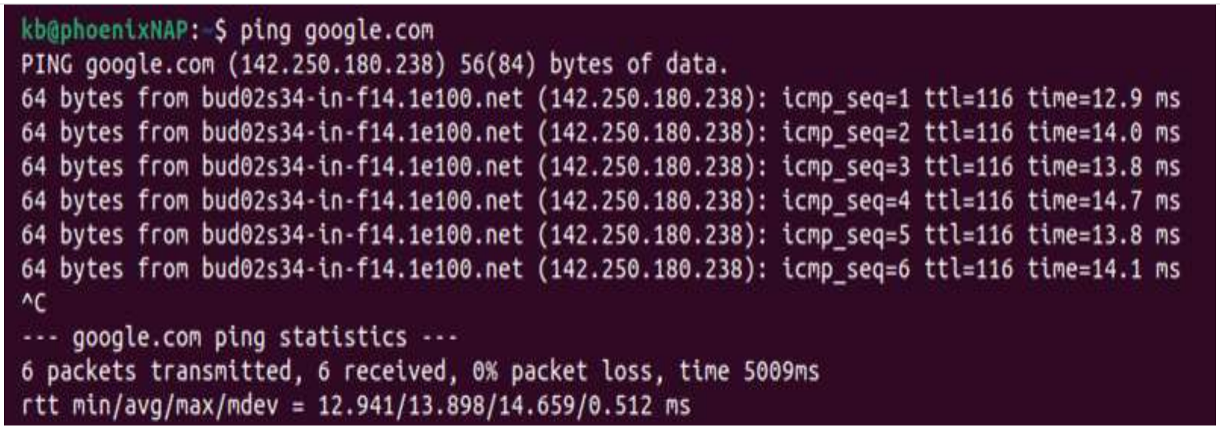
- **-15** saves all progress before closing the process.
- **-9** forces a stop immediately.

The process ID (PID) is unique for every program. Use the **ps** or **top** command to find the PID of a process.

26. ping command

Use the ping command (**p**acket **in**ternet **g**roper) to check internet connectivity. The tool is valuable in troubleshooting networking issues. Add an address to test how it works, for example:

```
ping google.com
```



```
kb@phoenixNAP:~$ ping google.com
PING google.com (142.250.180.238) 56(84) bytes of data.
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=1 ttl=116 time=12.9 ms
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=2 ttl=116 time=14.0 ms
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=3 ttl=116 time=13.8 ms
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=4 ttl=116 time=14.7 ms
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=5 ttl=116 time=13.8 ms
64 bytes from bud02s34-in-f14.1e100.net (142.250.180.238): icmp_seq=6 ttl=116 time=14.1 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 12.941/13.898/14.659/0.512 ms
```

The output shows the response time from the website. Press **CTRL**+**C** to stop the ping. If no response shows, there's a problem connecting to the host.

27. wget command

The wget command (WWW **get**) is used to download files from the internet. Use the following syntax to download a file:

```
wget <URL>
```

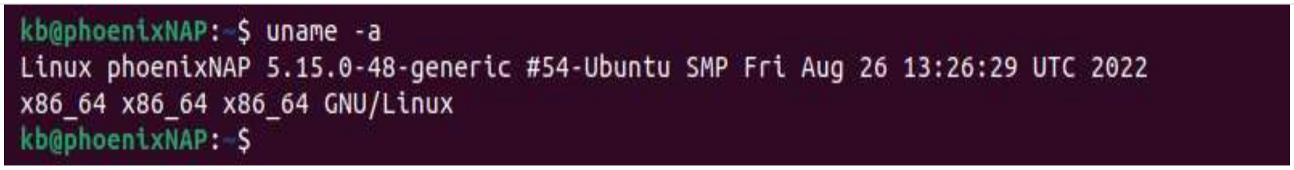
The command is robust and can continue downloads in unstable and slow networks.

Note: Learn how to solve wget: command not found error.

28. uname command

Use the uname command (**U**nix **name**) to print system information. Add the **-a** option to print a complete overview:

```
uname -a
```



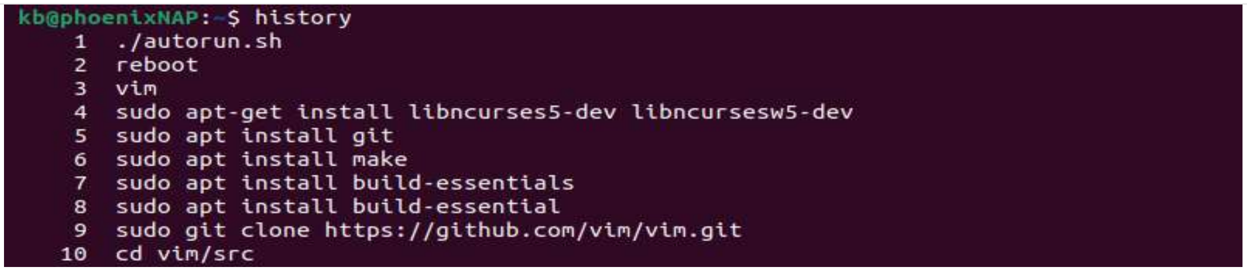
```
kb@phoenixNAP:~$ uname -a
Linux phoenixNAP 5.15.0-48-generic #54-Ubuntu SMP Fri Aug 26 13:26:29 UTC 2022
x86_64 x86_64 x86_64 GNU/Linux
kb@phoenixNAP:~$
```

The output shows the kernel version, OS, processor type, and other helpful information about the system.

29. history command

The terminal session keeps a history log of commands. To see the list, use the history command:

```
history
```



```
kb@phoenixNAP:~$ history
    1  ./autorun.sh
    2  reboot
    3  vim
    4  sudo apt-get install libncurses5-dev libncursesw5-dev
    5  sudo apt install git
    6  sudo apt install make
    7  sudo apt install build-essentials
    8  sudo apt install build-essential
    9  sudo git clone https://github.com/vim/vim.git
   10  cd vim/src
```

Add a number after the command to limit the number of entries if the list is long.

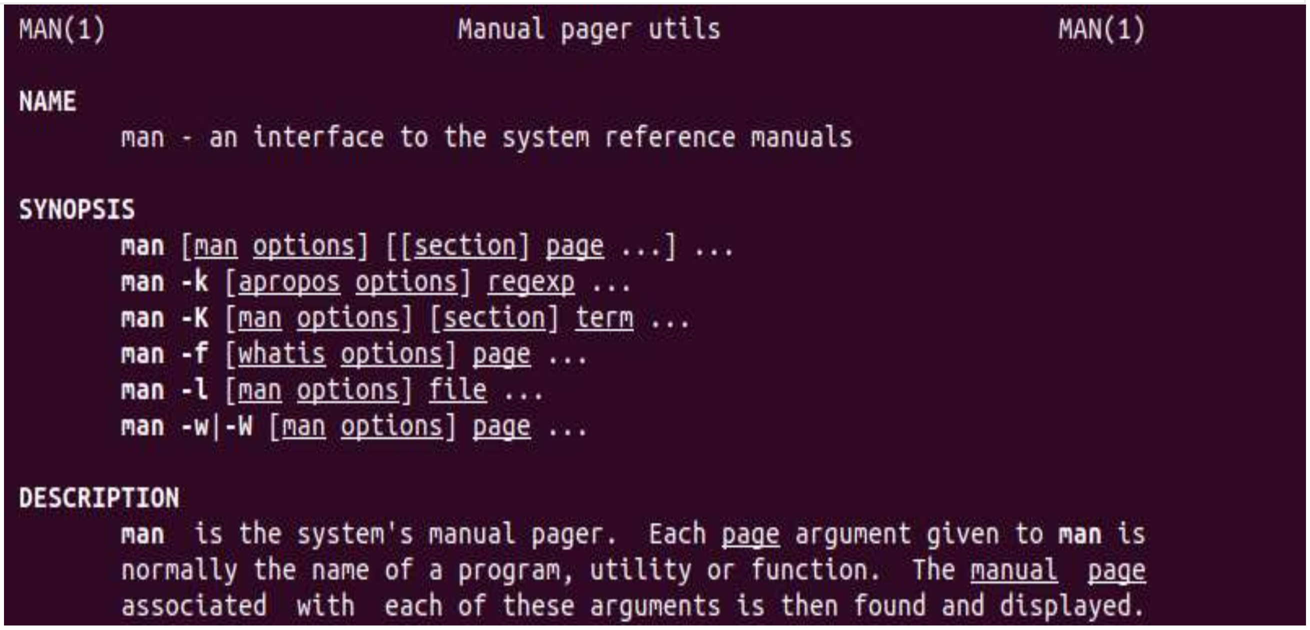
30. man command

The man command (**man**ual) is a convenient manual available in the terminal. Add **man** as a prefix to any command to check the manual reference:

```
man <command>
```

For example, to check the manual for the **man** command, run:

```
man man
```



```
MAN(1)                    Manual pager utils                    MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is the system's manual pager.  Each page argument given to man is
       normally the name of a program, utility or function.  The manual  page
       associated  with  each of these arguments is then found and displayed.
```

To exit the manual, press **q**.

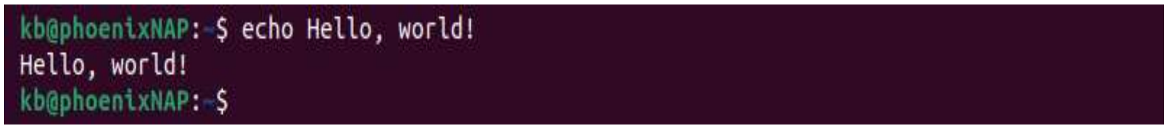
31. echo command

Use the echo command to print arguments to the terminal. The syntax is:

```
echo <argument>
```

For example, to print **Hello, world!** to the terminal run:

```
echo Hello, world!
```



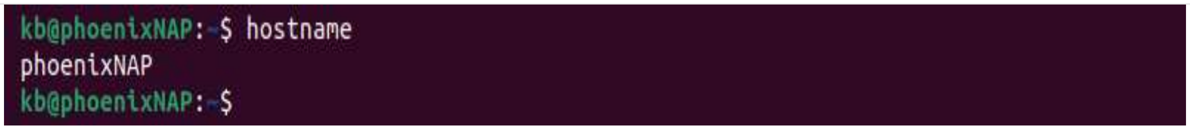
```
kb@phoenixNAP:~$ echo Hello, world!
Hello, world!
kb@phoenixNAP:~$
```

The command helps append text to files, print program results, and display Linux environment variables.

32. hostname command

To check the DNS name of the current machine, use the hostname command:

```
hostname
```



```
kb@phoenixNAP:~$ hostname
phoenixNAP
kb@phoenixNAP:~$
```

The hostname shows in the terminal as a result. Advanced features include changing the hostname, viewing and changing the system's domain, and checking the IP address.

Result

The basics and status of Linux commands were studied completely.

Ex. No.: 5b STUDY BASIC AND USER STATUS UNIX COMMANDS

AIM:

To study the basics of unix commands.

DEFINITIONS /THEORY:

Unix commands are a set of commands that are used to interact with the Unix operating system. Unix is a powerful, multi-user, multi-tasking operating system that was developed in the 1960s by Bell Labs. Unix commands are entered at the command prompt in a terminal window, and they allow users to perform a wide variety of tasks, such as managing files and directories, running processes, managing user accounts, and configuring network settings. Unix is now one of the most commonly used Operating systems used for various purposes such as Personal use, Servers, Smartphones, and many more. It was developed in the 1970's at AT & T Labs by two famous personalities Dennis M. Ritchie and Ken Thompson.

- You'll be surprised to know that the most popular programming language C came into existence to write the Unix Operating System.

- **Linux is Unix-Like operating system.**

- The most important part of the Linux is Linux Kernel which was first released in the early 90s by Linus Torvalds. There are several Linux distros available (most are open-source and free to download and use) such as Ubuntu, Debian, Fedora, Kali, Mint, Gentoo, Arch and much more.
- Now coming to the Basic and most usable commands of Linux/Unix part. (Please note that all the linux/unix commands are run in the terminal of a linux system.Terminal is like command prompt as that of in Windows OS)
- Linux/Unix commands are **case-sensitive** i.e Hello is different from hello**.**

Basic Unix commands:
- File System Navigation Unix Command
- File Manipulation Unix Command
- Process Management Unix Command
- Text Processing Unix Command
- Network Communication Unix Command
- Text Editors in Unix

File System Navigation Unix Command

Command	Description	Example
cd	Changes the current working directory.	cd Documents

Command	Description	Example
ls	Lists files and directories in the current directory.	ls
pwd	Prints the current working directory.	pwd
mkdir	Creates a new directory.	mkdir new_folder
rmdir	Removes an empty directory.	rmdir empty_folder
mv	Moves files or directories.	mv file1.txt Documents/

File Manipulation Unix Command

Command	Description	Example
touch	Creates an empty file or updates the access and modification times.	touch new_file.txt
cp	Copies files or directories.	cp file1.txt file2.txt
mv	Moves files or directories.	mv file1.txt Documents
rm	Remove files or directories.	rm old_file.txt
chmod	Changes the permissions of a file or directory.	chmod 644 file.txt
chown	Changes the owner and group of a file or directory.	chown user:group file.txt
ln	Creates links between files.	ln -s target_file symlink
cat	Concatenates files and displays their contents.	cat file1.txt file2.txt

Command	Description	Example
head	Displays the first few lines of a file.	head file.txt
tail	Displays the last few lines of a file.	tail file.txt
more	Displays the contents of a file page by page.	more file.txt
less	Displays the contents of a file with advanced navigation features.	less file.txt
diff	Compares files line by line.	diff file1.txt file2.txt
patch	Applies a diff file to update a target file.	patch file.txt < changes.diff

Process Management Unix Command

Command	Description	Example
ps	Displays information about active processes, including their status and IDs.	ps aux
top	Displays a dynamic real-time view of system processes and their resource usage.	top
kill	Terminates processes using their process IDs (PIDs).	kill <pid>
pkill	Sends signals to processes based on name or other attributes.	pkill -9 firefox

Command	Description	Example
killall	Terminates processes by name.	killall -9 firefox
renice	Changes the priority of running processes.	renice -n 10 <pid>
nice	Runs a command with modified scheduling priority.	nice -n 10 command
pstree	Displays running processes as a tree.	pstree
pgrep	Searches for processes by name or other attributes.	pgrep firefox
jobs	Lists active jobs and their status in the current shell session.	jobs
bg	Puts a job in the background.	bg <job_id>
fg	Brings a background job to the foreground.	fg <job_id>
nohup	Runs a command immune to hangups, with output to a specified file.	nohup command &
disown	Removes jobs from the shell’s job table, allowing them to run independently.	disown <job_id>

Text Processing Unix Command

Command	Description	Example

Command	Description	Example
grep	Searches for patterns in text files.	grep "error" logfile.txt
sed	Processes and transforms text streams.	sed 's/old_string/new_string/g' file.txt
awk	Processes and analyzes text files using a pattern scanning and processing language.	awk '{print $1, $3}' data.csv

Network Communication Unix Command

Command	Description	Example
ping	Tests connectivity with another host using ICMP echo requests.	ping google.com
traceroute	Traces the route that packets take to reach a destination.	traceroute google.com
nslookup	Queries DNS servers for domain name resolution and IP address information.	nslookup google.com
dig	Performs DNS queries, providing detailed information about DNS records.	dig google.com
host	Performs DNS lookups, displaying domain name to IP address resolution.	host google.com

Command	**Description**	**Example**
whois	Retrieves information about domain registration and ownership.	whois google.com
ssh	Provides secure remote access to a system.	ssh username@hostname
scp	Securely copies files between hosts over a network.	scp file.txt username@hostname:/path/
ftp	Transfers files between hosts using the File Transfer Protocol (FTP).	ftp hostname
telnet	Establishes interactive text-based communication with a remote host.	telnet hostname
netstat	Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.	netstat -tuln
ifconfig	Displays or configures network interfaces and their settings.	ifconfig
iwconfig	Configures wireless network interfaces.	iwconfig wlan0
route	Displays or modifies the IP routing table.	route -n

Command	Description	Example
arp	Displays or modifies the Address Resolution Protocol (ARP) cache.	arp -a
ss	Displays socket statistics.	ss -tuln
hostname	Displays or sets the system’s hostname.	hostname
mtr	Combines the functionality of ping and traceroute, providing detailed network diagnostic information.	mtr google.com

System Administration Unix Command

Command	Description	Example
df	Displays disk space usage.	df -h
du	Displays disk usage of files and directories.	du -sh /path/to/directory
crontab -e	Manages cron jobs, which are scheduled tasks that run at predefined times or intervals.	crontab -e

Text Editors in Unix

Text Editor	Description	Example
Vi / Vim	Vi (Vim) is a highly configurable, powerful, and feature-rich text editor based on the original Vi editor. Vim offers modes for both command-line operations and text editing.	Open a file with Vim: vim filename Exit Vim editor: Press Esc, then type :wq and press Enter

Text Editor	**Description**	**Example**
Emacs	Emacs is a versatile text editor with extensive customization capabilities and support for various programming languages.	Open a file with Emacs: emacs filename Save and exit Emacs: Press Ctrl + X, then Ctrl + S and Ctrl + X, then Ctrl + C to exit
Nano	Nano is a simple and user-friendly text editor designed for ease of use and accessibility.	Open a file with Nano: nano filename Save and exit Nano: Press Ctrl + O, then Ctrl + X
Ed	Ed is a standard Unix text editor that operates in line-oriented mode, making it suitable for batch processing and automation tasks.	Open a file with Ed: ed filename Exit Ed editor: Type q and press Enter
Jed	Jed is a lightweight yet powerful text editor that provides an intuitive interface and support for various programming languages.	Open a file with Jed: jed filename Save and exit Jed: Press Alt + X, then type exit and press Enter

Inference:

Result

The basics and status of unix commands were studied completely.