Practical File
Of
Operating System
22CS005

Submitted

in partial fulfillment for the award of the degree of

BACHELEOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

CHITKARA UNIVERSITY

CHANDIGARH-PATIALA NATIONAL HIGHWAY
RAJPURA (PATIALA) PUNJAB-140401 (INDIA)

April, 2024

Submitted To:	**Submitted By:**
Dr. SAVITA WADHAWAN	Deepanshu Singla
Assistant Professor	2310991645
Chitkara University, Punjab	2nd Sem,2023

Index

Program-5

Aim : Implement the basic and user status commands like: su, sudo, man, help, history, who, whoami, id, uname, uptime, free, tty, cal, date, hostname, reboot, clear.

Solution:

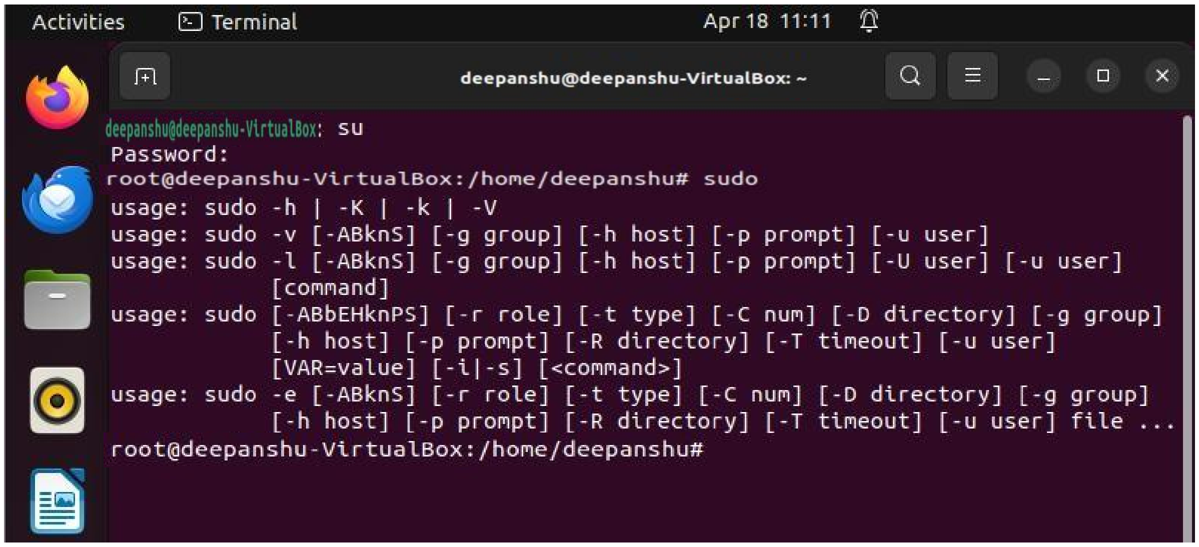
User status commands are commands in a Unix-based operating system that provide information about the current user and their privileges, system resources, and system status. These commands are used to manage users, processes, and system resources, and they are typically run from the command line or terminal window. Some common user status commands include su, sudo, whoami, id, uptime, free, and tty. These commands provide information about the current user's identity, permissions, and system activity. They are useful for system administration and troubleshooting, as well as for general system maintenance and monitoring. Some of the commonly used user status commands include:

- **su:** su (short for substitute user) is a command used to switch the current user identity to any other user identity, including root (the superuser). By default, su command switches to the root user account, but you can specify the username of the user account you want to switch to as an argument.
- **sudo:** sudo (short for SuperUser DO) is a command that allows a user with administrative privileges to execute commands as another user, such as the root user. The sudo command requires users to enter their own password to perform the command, as opposed to the root user's password.
- **man:** man (short for manual) is a command that displays the user manual for a specific command. It is used to get information about the syntax, usage, and options available for a particular command.
- **help:** help is a command that displays built-in help for the shell in use, including a list of all available commands and their descriptions.
- **history:** history is a command that displays a list of the previously executed commands in the terminal session, along with their sequence numbers.
- **who:** who is a command that displays information about all currently logged-in users on the system, including their usernames, terminal sessions, and login times.
- **whoami:** whoami is a command that displays the current username of the user running the command.

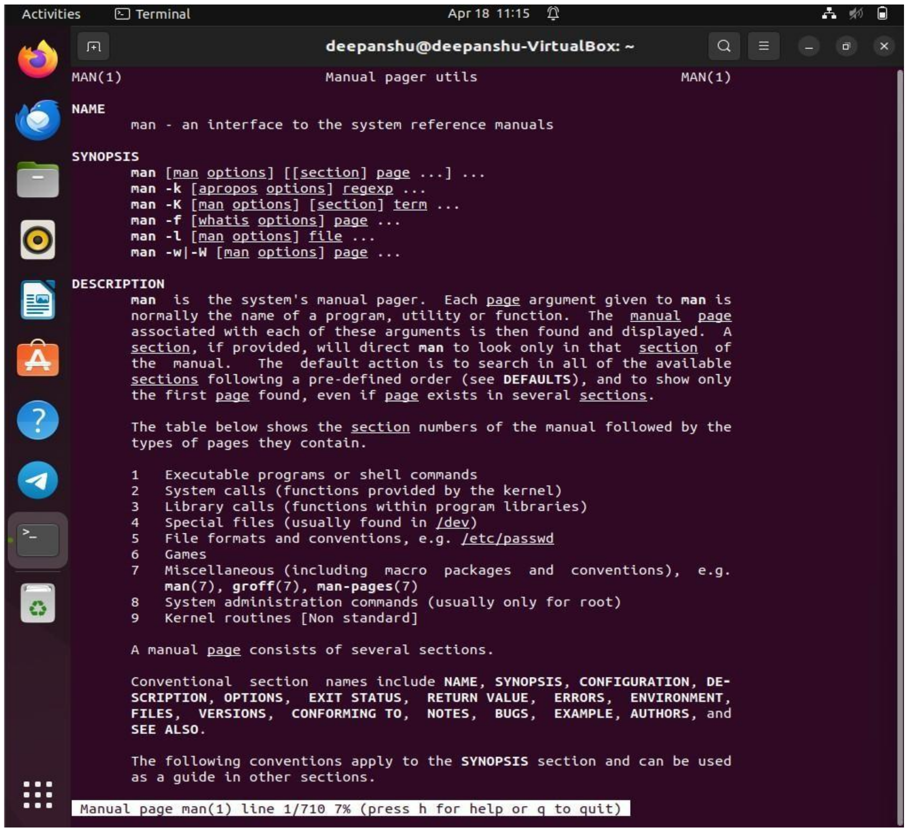
- **id:** id is a command that displays the current user's UID (User ID), GID (Group ID), and group memberships.
- **uname:** uname is a command that displays system information about the operating system and its kernel, such as the system name, version, and release.
- **uptime:** uptime is a command that displays the current system uptime, which is the time elapsed since the system was last booted.
- **free:** free is a command that displays information about the system's memory usage, including the total, used, and free memory available.
- **tty:** tty is a command that displays the terminal device name associated with the current terminal session.
- **cal:** cal is a command that displays a calendar for the current or specified month or year.
- **date:** date is a command that displays the current system date and time in the specified format.
- **hostname:** hostname is a command that displays the current hostname of the system.
- **reboot:** reboot is a command that is used to reboot the system.
- **clear:** clear is a command that clears the terminal screen, removing all previous output from the screen.

Execution of these commands-

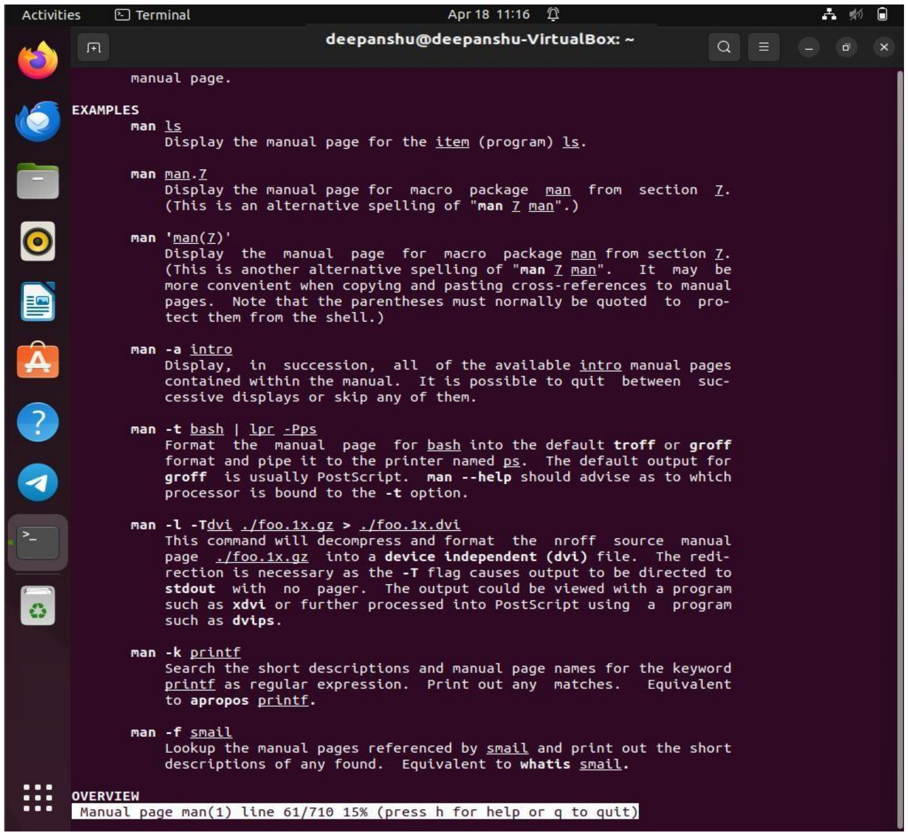
Open the terminal in the linux and these commands will be executed there.



```
deepanshu@deepanshu-VirtualBox: su
Password:
root@deepanshu-VirtualBox:/home/deepanshu# sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-ABknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-ABbEHknPS] [-r role] [-t type] [-C num] [-D directory] [-g group]
            [-h host] [-p prompt] [-R directory] [-T timeout] [-u user]
            [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-ABknS] [-r role] [-t type] [-C num] [-D directory] [-g group]
            [-h host] [-p prompt] [-R directory] [-T timeout] [-u user] file ...
root@deepanshu-VirtualBox:/home/deepanshu#
```

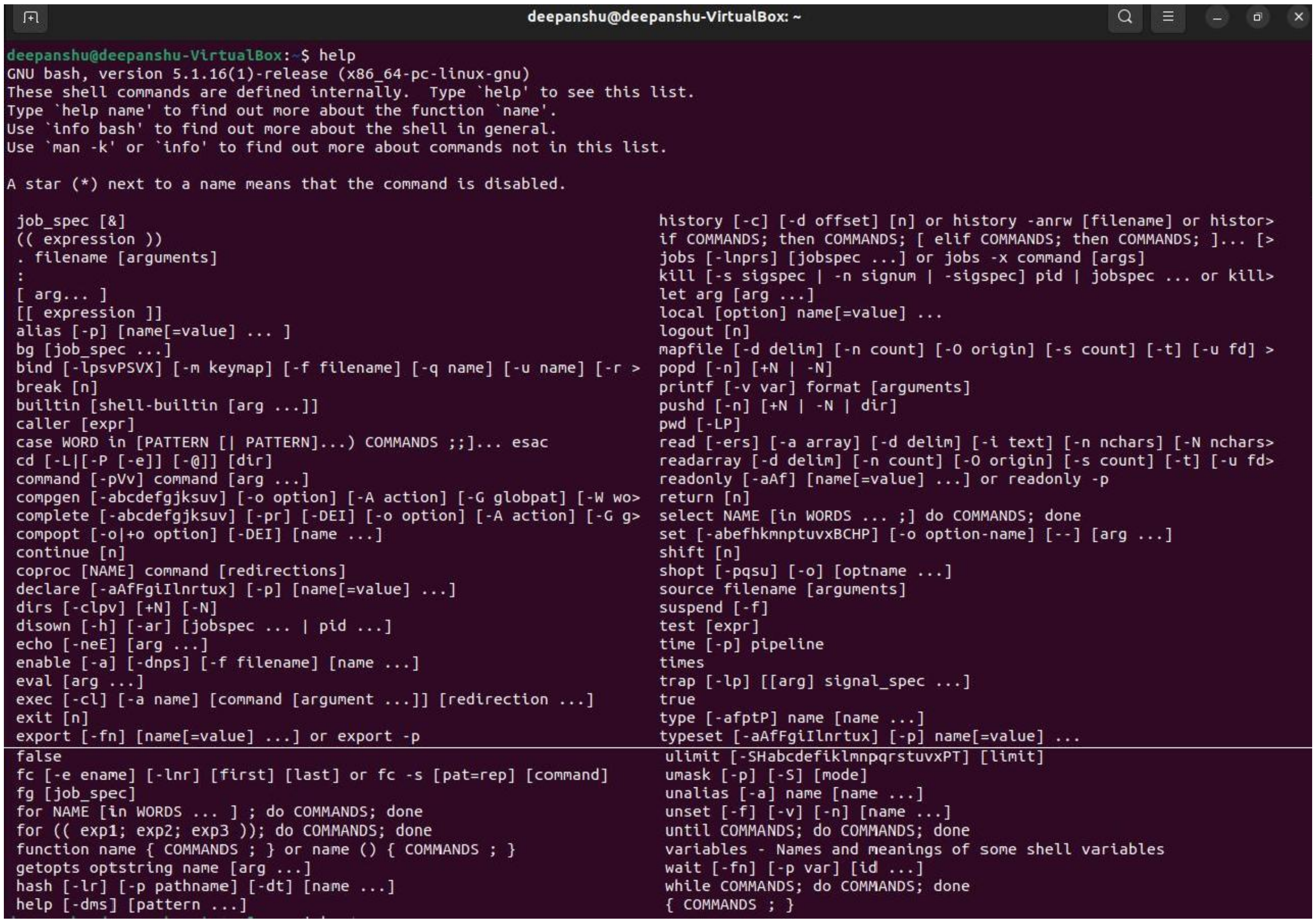


```
MAN(1)                        Manual pager utils                        MAN(1)

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.  A
       section, if provided, will direct man to look only in that  section  of
       the  manual.   The  default action is to search in all of the available
       sections following a pre-defined order (see DEFAULTS), and to show only
       the first page found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the
       types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous (including  macro  packages  and  conventions),  e.g.
           man(7), groff(7), man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several sections.

       Conventional  section  names include NAME, SYNOPSIS, CONFIGURATION, DE-
       SCRIPTION, OPTIONS,  EXIT STATUS,  RETURN VALUE,  ERRORS,  ENVIRONMENT,
       FILES,  VERSIONS,  CONFORMING TO,  NOTES,  BUGS,  EXAMPLE, AUTHORS, and
       SEE ALSO.

       The following conventions apply to the SYNOPSIS section and can be used
       as a guide in other sections.

 Manual page man(1) line 1/710 7% (press h for help or q to quit)
```

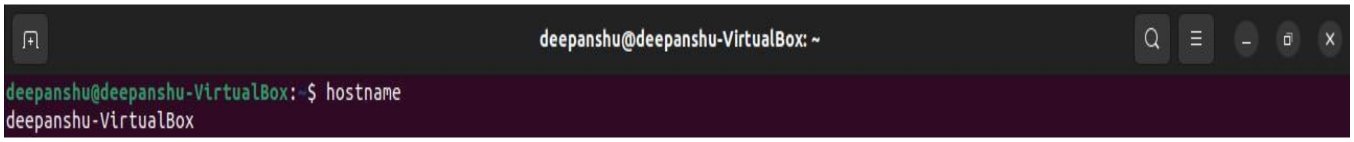


```
       manual page.

EXAMPLES
       man ls
           Display the manual page for the item (program) ls.

       man man.7
           Display the manual page for  macro  package  man  from  section  7.
           (This is an alternative spelling of "man 7 man".)

       man 'man(7)'
           Display  the  manual  page  for  macro  package man from section 7.
           (This is another alternative spelling of "man 7 man".   It  may  be
           more convenient when copying and pasting cross-references to manual
           pages.  Note that the parentheses must normally be quoted  to  pro-
           tect them from the shell.)

       man -a intro
           Display,  in  succession,  all  of the available intro manual pages
           contained within the manual.  It is possible to quit  between  suc-
           cessive displays or skip any of them.

       man -t bash | lpr -Pps
           Format  the  manual  page  for bash into the default troff or groff
           format and pipe it to the printer named ps.  The default output for
           groff  is usually PostScript.  man --help should advise as to which
           processor is bound to the -t option.

       man -l -Tdvi ./foo.1x.gz > ./foo.1x.dvi
           This command will decompress and format  the  nroff  source  manual
           page  ./foo.1x.gz  into a device independent (dvi) file.  The redi-
           rection is necessary as the -T flag causes output to be directed to
           stdout  with  no  pager.  The output could be viewed with a program
           such as xdvi or further processed into PostScript using  a  program
           such as dvips.

       man -k printf
           Search the short descriptions and manual page names for the keyword
           printf as regular expression.  Print out any  matches.   Equivalent
           to apropos printf.

       man -f smail
           Lookup the manual pages referenced by smail and print out the short
           descriptions of any found.  Equivalent to whatis smail.

OVERVIEW
 Manual page man(1) line 61/710 15% (press h for help or q to quit)
```

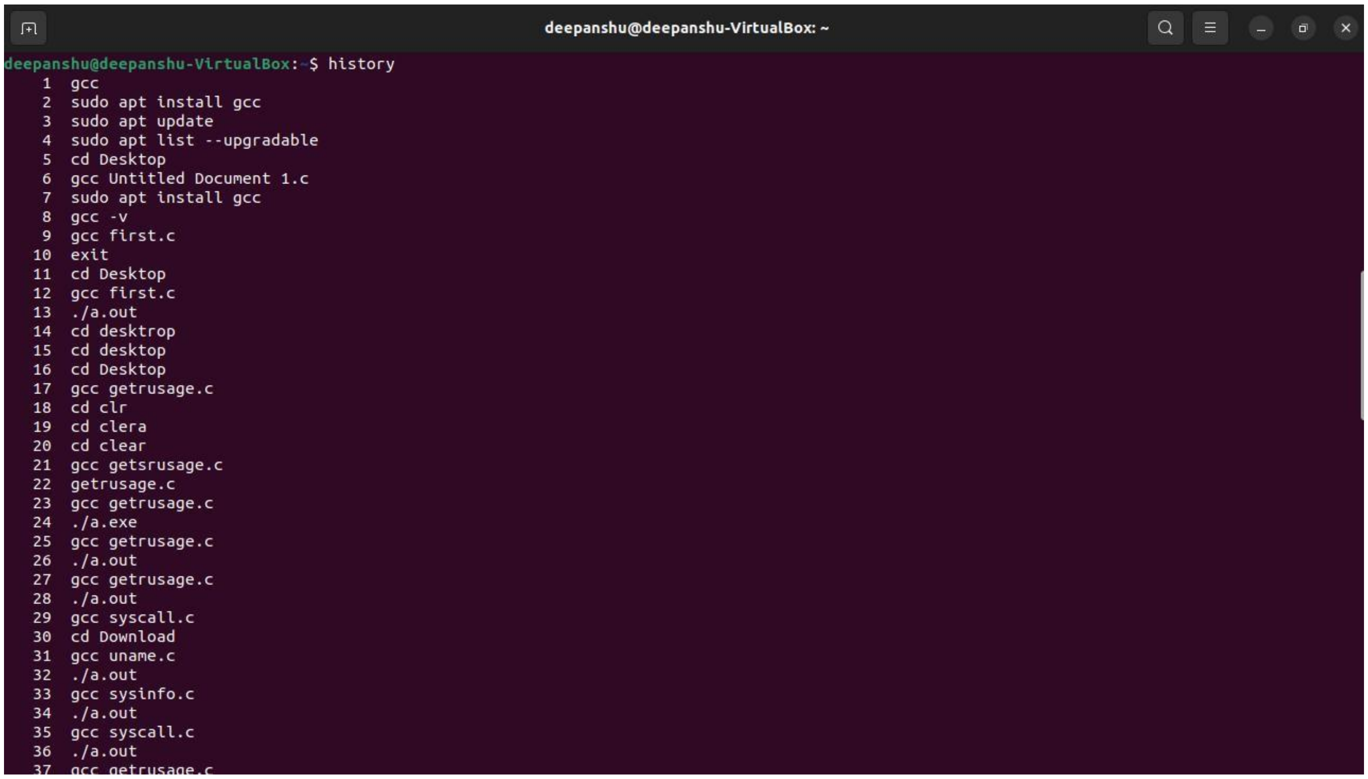
(By running man the above output came)



```
deepanshu@deepanshu-VirtualBox:~$ help
GNU bash, version 5.1.16(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                                                          history [-c] [-d offset] [n] or history -anrw [filename] or histor>
 (( expression ))                                                      if COMMANDS; then COMMANDS; [ elif COMMANDS; then COMMANDS; ]... [>
 . filename [arguments]                                                jobs [-lnprs] [jobspec ...] or jobs -x command [args]
 :                                                                     kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill>
 [ arg... ]                                                            let arg [arg ...]
 [[ expression ]]                                                      local [option] name[=value] ...
 alias [-p] [name[=value] ... ]                                        logout [n]
 bg [job_spec ...]                                                     mapfile [-d delim] [-n count] [-O origin] [-s count] [-t] [-u fd] >
 bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u name] [-r >  popd [-n] [+N | -N]
 break [n]                                                             printf [-v var] format [arguments]
 builtin [shell-builtin [arg ...]]                                     pushd [-n] [+N | -N | dir]
 caller [expr]                                                         pwd [-LP]
 case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;]... esac            read [-ers] [-a array] [-d delim] [-i text] [-n nchars] [-N nchars>
 cd [-L|[-P [-e]] [-@]] [dir]                                          readarray [-d delim] [-n count] [-O origin] [-s count] [-t] [-u fd>
 command [-pVv] command [arg ...]                                      readonly [-aAf] [name[=value] ...] or readonly -p
 compgen [-abcdefgjksuv] [-o option] [-A action] [-G globpat] [-W wo>  return [n]
 complete [-abcdefgjksuv] [-pr] [-DEI] [-o option] [-A action] [-G g>  select NAME [in WORDS ... ;] do COMMANDS; done
 compopt [-o|+o option] [-DEI] [name ...]                              set [-abefhkmnptuvxBCHP] [-o option-name] [--] [arg ...]
 continue [n]                                                          shift [n]
 coproc [NAME] command [redirections]                                  shopt [-pqsu] [-o] [optname ...]
 declare [-aAfFgiIlnrtux] [-p] [name[=value] ...]                      source filename [arguments]
 dirs [-clpv] [+N] [-N]                                                suspend [-f]
 disown [-h] [-ar] [jobspec ... | pid ...]                             test [expr]
 echo [-neE] [arg ...]                                                 time [-p] pipeline
 enable [-a] [-dnps] [-f filename] [name ...]                          times
 eval [arg ...]                                                        trap [-lp] [[arg] signal_spec ...]
 exec [-cl] [-a name] [command [argument ...]] [redirection ...]       true
 exit [n]                                                              type [-afptP] name [name ...]
 export [-fn] [name[=value] ...] or export -p                          typeset [-aAfFgiIlnrtux] [-p] name[=value] ...
 false                                                                 ulimit [-SHabcdefiklmnpqrstuvxPT] [limit]
 fc [-e ename] [-lnr] [first] [last] or fc -s [pat=rep] [command]      umask [-p] [-S] [mode]
 fg [job_spec]                                                         unalias [-a] name [name ...]
 for NAME [in WORDS ... ] ; do COMMANDS; done                          unset [-f] [-v] [-n] [name ...]
 for (( exp1; exp2; exp3 )); do COMMANDS; done                         until COMMANDS; do COMMANDS; done
 function name { COMMANDS ; } or name () { COMMANDS ; }                variables - Names and meanings of some shell variables
 getopts optstring name [arg ...]                                      wait [-fn] [-p var] [id ...]
 hash [-lr] [-p pathname] [-dt] [name ...]                             while COMMANDS; do COMMANDS; done
 help [-dms] [pattern ...]                                             { COMMANDS ; }
```

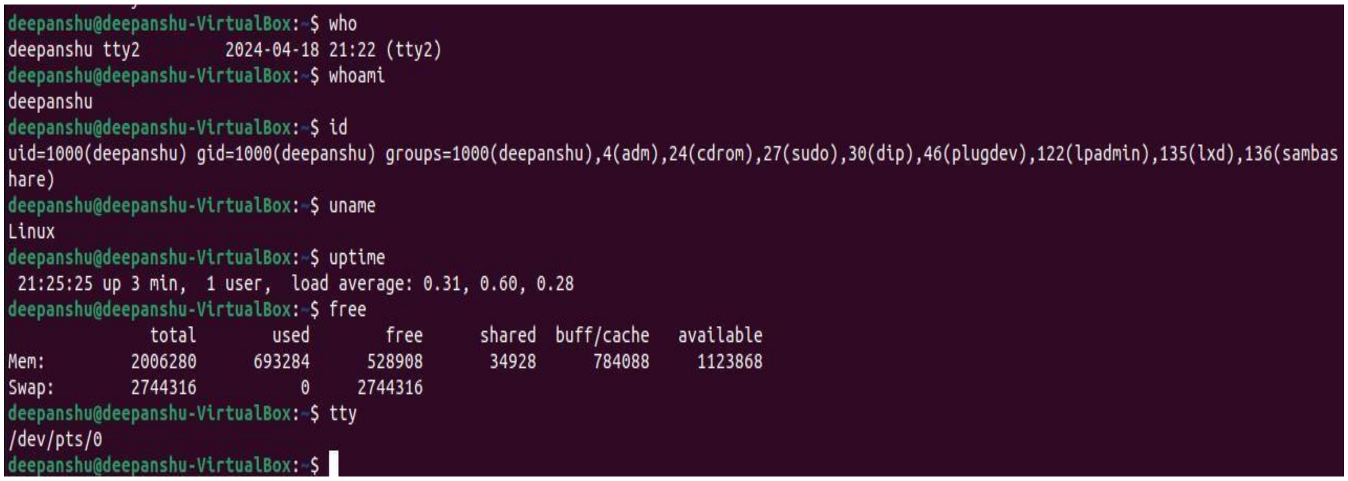


```
deepanshu@deepanshu-VirtualBox:~$ hostname
deepanshu-VirtualBox
```



deepanshu@deepanshu-VirtualBox: ~

```
deepanshu@deepanshu-VirtualBox:~$ history
    1  gcc
    2  sudo apt install gcc
    3  sudo apt update
    4  sudo apt list --upgradable
    5  cd Desktop
    6  gcc Untitled Document 1.c
    7  sudo apt install gcc
    8  gcc -v
    9  gcc first.c
   10  exit
   11  cd Desktop
   12  gcc first.c
   13  ./a.out
   14  cd desktrop
   15  cd desktop
   16  cd Desktop
   17  gcc getrusage.c
   18  cd clr
   19  cd clera
   20  cd clear
   21  gcc getsrusage.c
   22  getrusage.c
   23  gcc getrusage.c
   24  ./a.exe
   25  gcc getrusage.c
   26  ./a.out
   27  gcc getrusage.c
   28  ./a.out
   29  gcc syscall.c
   30  cd Download
   31  gcc uname.c
   32  ./a.out
   33  gcc sysinfo.c
   34  ./a.out
   35  gcc syscall.c
   36  ./a.out
   37  gcc getrusage.c
```



```
deepanshu@deepanshu-VirtualBox:~$ who
deepanshu tty2         2024-04-18 21:22 (tty2)
deepanshu@deepanshu-VirtualBox:~$ whoami
deepanshu
deepanshu@deepanshu-VirtualBox:~$ id
uid=1000(deepanshu) gid=1000(deepanshu) groups=1000(deepanshu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),122(lpadmin),135(lxd),136(sambas
hare)
deepanshu@deepanshu-VirtualBox:~$ uname
Linux
deepanshu@deepanshu-VirtualBox:~$ uptime
 21:25:25 up 3 min,  1 user,  load average: 0.31, 0.60, 0.28
deepanshu@deepanshu-VirtualBox:~$ free
               total        used        free      shared  buff/cache   available
Mem:         2006280      693284      528908       34928      784088     1123868
Swap:        2744316           0     2744316
deepanshu@deepanshu-VirtualBox:~$ tty
/dev/pts/0
deepanshu@deepanshu-VirtualBox:~$ 
```

Program-6

Aim : Implement deadlock in C by using shared variable.

Solution:

Deadlock occurs when two or more processes are unable to proceed because each is waiting for a resource that is being held by another process. In the context of shared variables, deadlock can be created by introducing a circular dependency between processes, where each process holds a resource that the other process needs. Some key concepts of deadlock:

1. Resources: Resources are entities used by processes during their execution. They can be classified as preemptable (can be taken away from a process) or non- preemptable (cannot be forcibly taken away). Examples include CPU time, memory, printers, and file systems.

2. Resource Allocation Graph (RAG): The RAG is a graphical representation of the resource allocation state in a system. It consists of process (P) nodes and resource (R) nodes, connected by directed edges indicating resource allocation and request relationships.

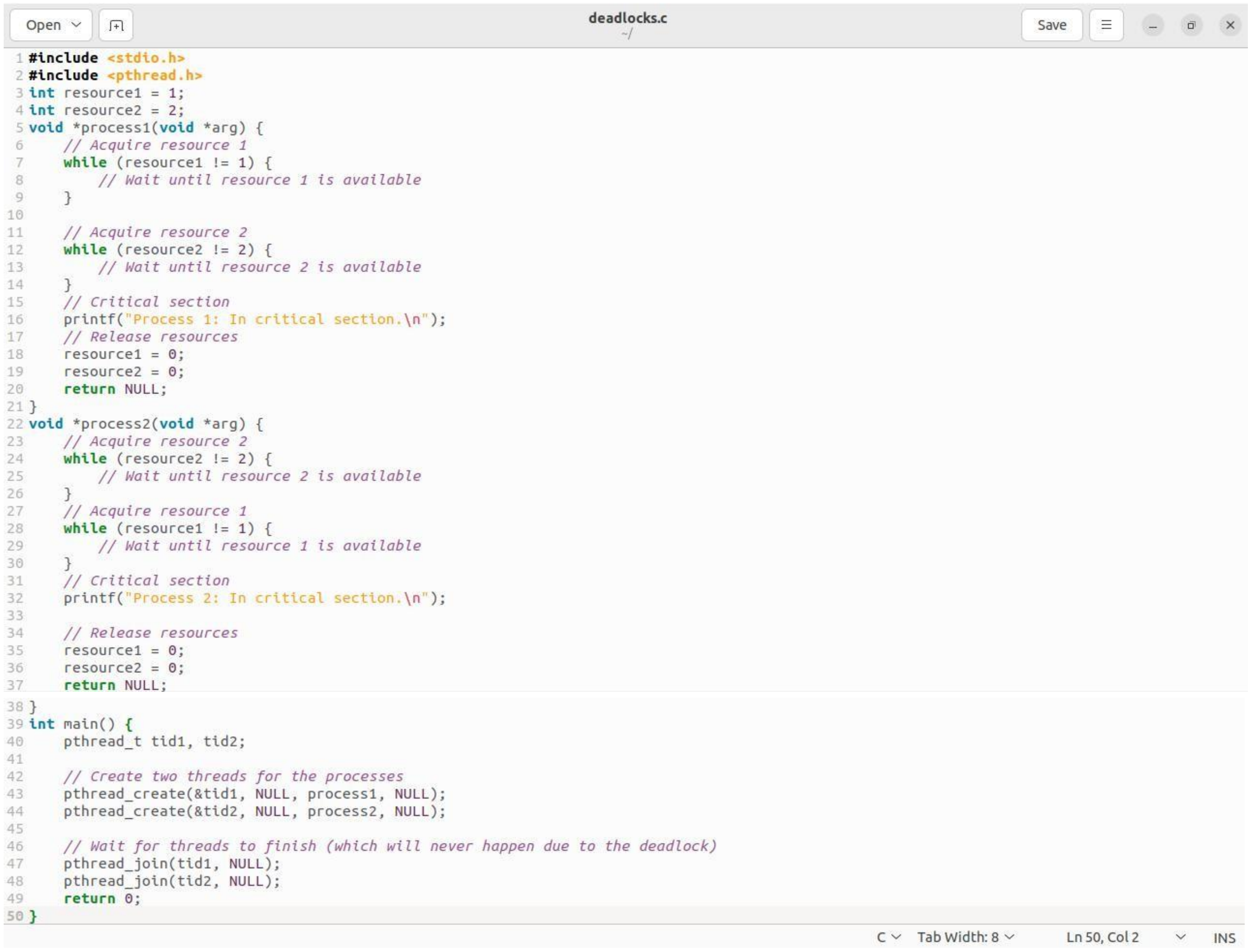
3. Conditions for Deadlock (Coffman conditions): Deadlocks occur when four necessary conditions are simultaneously satisfied:

    - Mutual Exclusion: At least one resource must be non-shareable, allowing only one process to use it at a time.
    - Hold and Wait: Processes hold allocated resources while requesting additional resources.
    - No Preemption: Resources cannot be forcibly taken away from a process; they can only be released voluntarily. d. Circular Wait: A circular chain of processes exists, where each process is waiting for a resource held by the next process in the chain.

4. Deadlock Prevention, Avoidance, and Detection:

    - Deadlock Prevention: Focuses on designing systems to prevent one or more Coffman conditions from occurring. However, prevention techniques often restrict system flexibility and resource utilization.
    - Deadlock Avoidance: Dynamically analyzes the resource allocation state to determine if a specific allocation will potentially lead to a deadlock. If a deadlock is predicted, the system avoids that allocation to ensure safety.
    - Deadlock Detection and Recovery: Periodically checks the resource allocation state to detect deadlocks. If a deadlock is detected, the system takes corrective actions such as killing processes or rolling back transactions to recover from the deadlock stat.

Program:

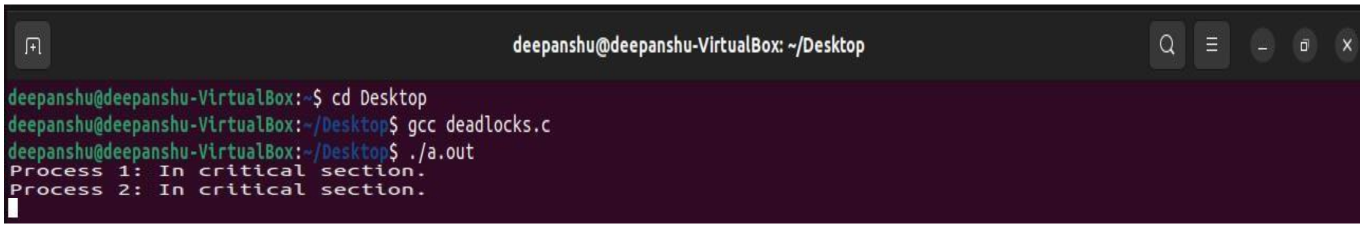


```c
#include <stdio.h>
#include <pthread.h>
int resource1 = 1;
int resource2 = 2;
void *process1(void *arg) {
    // Acquire resource 1
    while (resource1 != 1) {
        // Wait until resource 1 is available
    }

    // Acquire resource 2
    while (resource2 != 2) {
        // Wait until resource 2 is available
    }
    // Critical section
    printf("Process 1: In critical section.\n");
    // Release resources
    resource1 = 0;
    resource2 = 0;
    return NULL;
}
void *process2(void *arg) {
    // Acquire resource 2
    while (resource2 != 2) {
        // Wait until resource 2 is available
    }
    // Acquire resource 1
    while (resource1 != 1) {
        // Wait until resource 1 is available
    }
    // Critical section
    printf("Process 2: In critical section.\n");

    // Release resources
    resource1 = 0;
    resource2 = 0;
    return NULL;
}
int main() {
    pthread_t tid1, tid2;

    // Create two threads for the processes
    pthread_create(&tid1, NULL, process1, NULL);
    pthread_create(&tid2, NULL, process2, NULL);

    // Wait for threads to finish (which will never happen due to the deadlock)
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    return 0;
}
```

The program creates two threads representing two processes. Each process tries to acquire two shared resources in a specific order. However, due to the circular wait condition, where each process is waiting for the resource held by the other process, a deadlock occurs. The program then hangs indefinitely without making any progress.

Output :

The output of the code will depend on the scheduling of the threads and the timing of the resource availability. In some cases, you might observe the following output:



```
deepanshu@deepanshu-VirtualBox:~$ cd Desktop
deepanshu@deepanshu-VirtualBox:~/Desktop$ gcc deadlocks.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ ./a.out
Process 1: In critical section.
Process 2: In critical section.
```

The program will then hang indefinitely, as both threads are waiting for the resources held by the other thread. This situation leads to a deadlock, and the program will not proceed further or produce any additional output.
The exact output may vary depending on the operating system, thread scheduling, and timing of resource availability. In some cases, the output might be different, or the program may appear to be stuck without any output. The behavior is non-deterministic due to the nature of concurrent execution and the possibility of different thread interleavings.

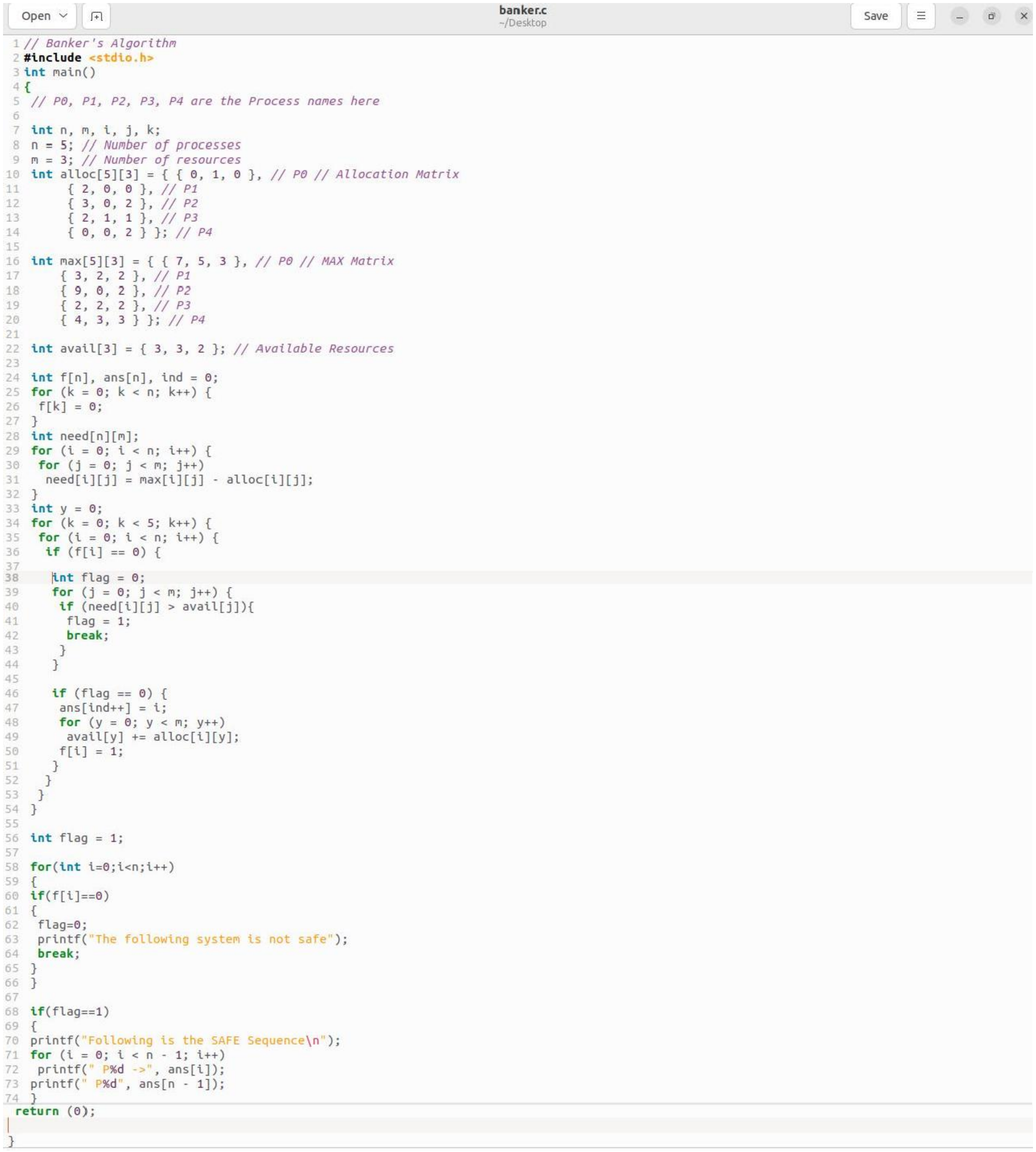
To Avoid deadlock , **Banker's algorithm** is one of the simplest known solutions to the mutual exclusion problem forthe general case of N process. Banker's Algorithm is a critical section solution for **N** processes. The algorithm preserves the first come first serve property.

How does the Banker's Algorithm work?
In the Banker's Algorithm, each process is assigned a number (a ticket) in a lexicographical order. Before entering the critical section, a process receives a ticket number, and the process with the smallest ticket number enters the critical section. If two processes receive the same ticket number, theprocess with the lower process ID is given priority.

Advantages of Bakery Algorithm:
- Fairness
- Easy to Implement
- No Deadlock
- No starvation

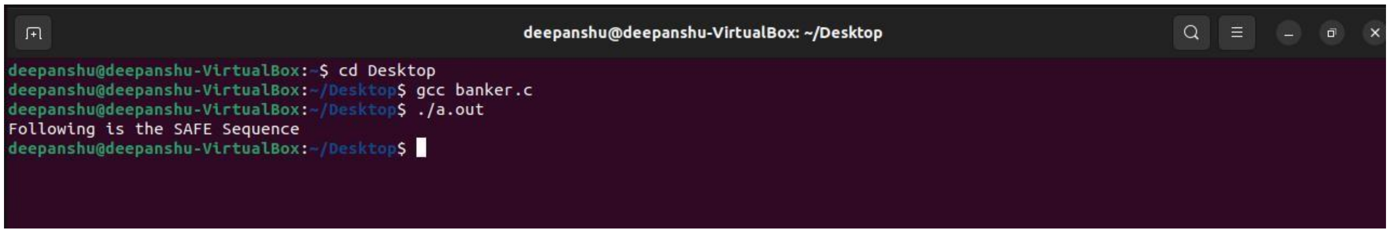
Disadvantages Bakery Algorithm:
- Not Scalable
- High Time Complexity
- Busy Waiting
- Memory Overhead

Program:



```c
// Banker's Algorithm
#include <stdio.h>
int main()
{
 // P0, P1, P2, P3, P4 are the Process names here

 int n, m, i, j, k;
 n = 5; // Number of processes
 m = 3; // Number of resources
 int alloc[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix
      { 2, 0, 0 }, // P1
      { 3, 0, 2 }, // P2
      { 2, 1, 1 }, // P3
      { 0, 0, 2 } }; // P4

 int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix
     { 3, 2, 2 }, // P1
     { 9, 0, 2 }, // P2
     { 2, 2, 2 }, // P3
     { 4, 3, 3 } }; // P4

 int avail[3] = { 3, 3, 2 }; // Available Resources

 int f[n], ans[n], ind = 0;
 for (k = 0; k < n; k++) {
  f[k] = 0;
 }
 int need[n][m];
 for (i = 0; i < n; i++) {
  for (j = 0; j < m; j++)
   need[i][j] = max[i][j] - alloc[i][j];
 }
 int y = 0;
 for (k = 0; k < 5; k++) {
  for (i = 0; i < n; i++) {
   if (f[i] == 0) {

    int flag = 0;
    for (j = 0; j < m; j++) {
     if (need[i][j] > avail[j]){
      flag = 1;
      break;
     }
    }

    if (flag == 0) {
     ans[ind++] = i;
     for (y = 0; y < m; y++)
      avail[y] += alloc[i][y];
     f[i] = 1;
    }
   }
  }
 }

 int flag = 1;

 for(int i=0;i<n;i++)
 {
 if(f[i]==0)
 {
  flag=0;
  printf("The following system is not safe");
  break;
 }
 }

 if(flag==1)
 {
 printf("Following is the SAFE Sequence\n");
 for (i = 0; i < n - 1; i++)
  printf(" P%d ->", ans[i]);
 printf(" P%d", ans[n - 1]);
 }
 return (0);

}
```

OUTPUT:



```
deepanshu@deepanshu-VirtualBox:~$ cd Desktop
deepanshu@deepanshu-VirtualBox:~/Desktop$ gcc banker.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ ./a.out
Following is the SAFE Sequence
deepanshu@deepanshu-VirtualBox:~/Desktop$
```

Program-7

Aim : Implement the commands that is used for Creating and Manipulating files: cat, cp, mv, rm, ls and its options, touch and their options, which is, where is, what is.

Solution:

1. **cat:** This command is used to display the contents of a file.
2. **cp:** The cp command is used to copy files and directories. Options:
   - -r or -R: Recursively copy directories and their contents.
3. **mv:** The mv command is used to move or rename files and directories. Options:
   - -i: Prompt before overwriting an existing file.
   - -u: Move only when the source file is newer than the destination file or when the destination file does not exist.
4. **rm:** The rm command is used to remove files and directories. Options:
   - -r or -R: Recursively remove directories and their contents.
   - -f: Force removal without prompting for confirmation.
5. **ls:** The ls command is used to list the files and directories in a directory. Options:
   - -l: Displays the long format listing, including file permissions, ownership, size, modification time, and other details.
   - -a: Shows all files, including hidden files that start with a dot.
   - -h: Prints file sizes in a human-readable format, such as "K" for kilobytes or "M" for megabytes.
   - -t: Sorts files by modification time, with the most recently modified files appearing first.
   - -r: Reverses the order of the file listing.
   - -S: Sorts files by size, with the largest files appearing first.
   - -i: Displays the inode number of each file.
   - -R: Recursively lists files and directories in subdirectories.
   - -g: Similar to the long format listing (-l), but does not display the owner of the file.
   - --color: Enables colorized output, making it easier to distinguish file types.

6. **touch:** The touch command is used to create an empty file or update the timestamp of an existing file.

   Options:

   - -a: Change only the access time.
   - -m: Change only the modification time.

7. **tac:** The tac command is the reverse of the cat command. It is used to display the contents of a file in reverse order, with the last line appearing first and the first line appearing last.

8. **rev:** The rev command is used to reverse the characters in each line of a file or standard input. It is particularly useful for reversing the order of characters within a line or reversing the contents of a file.

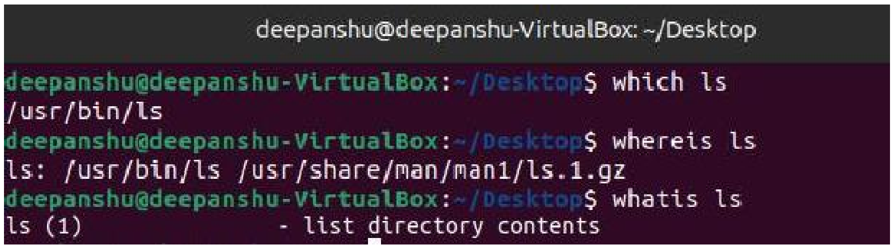
Additionally, here are some other commonly used commands:

1. **which:** The which command is used to locate the executable file that is associated with a given command.
2. **whereis:** The whereis command is used to locate the binary, source, and manual page files for a command.
3. **whatis:** The whatis command is used to display a brief description of a command.

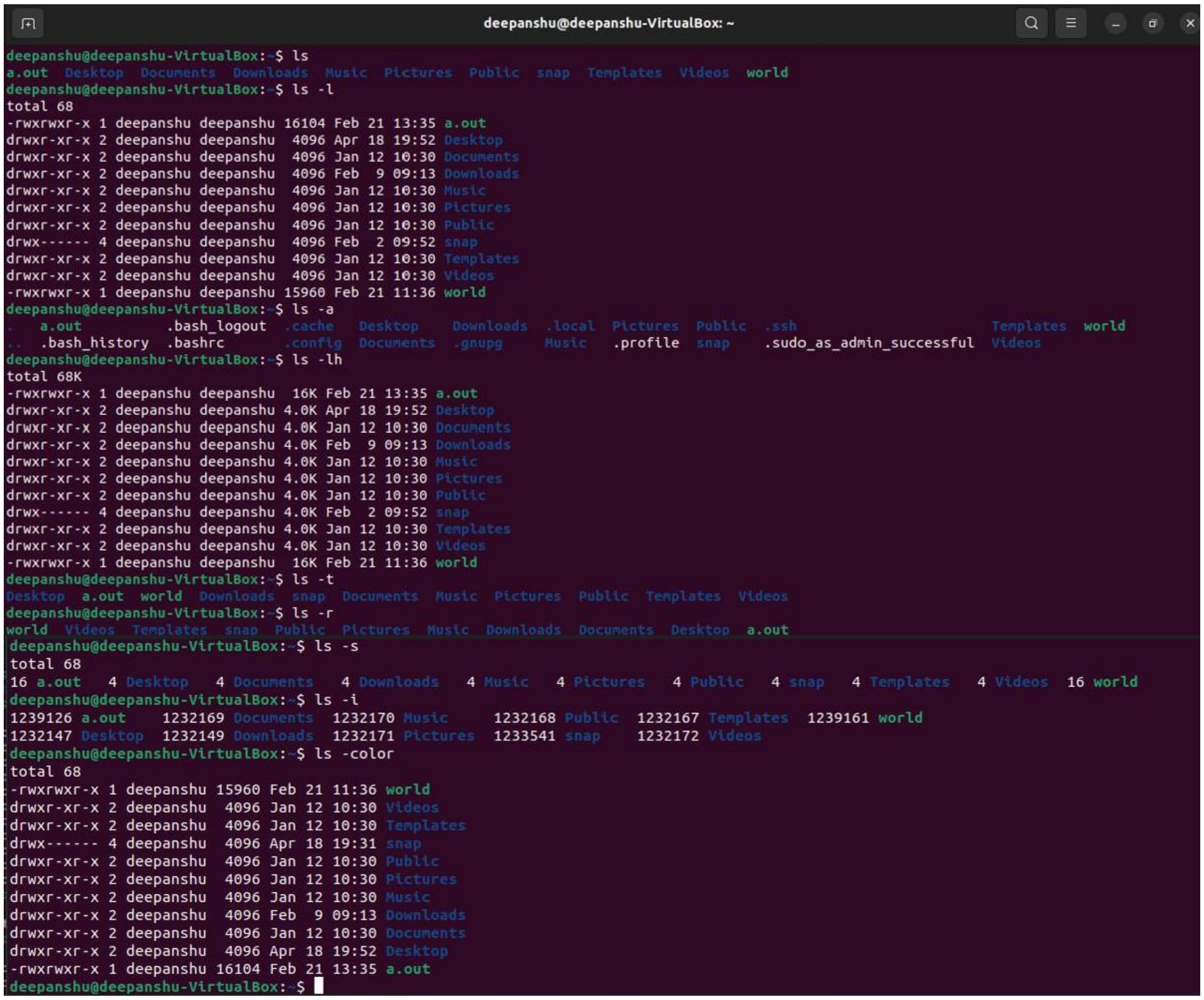
These commands and their options provide the basic functionality for creating, manipulating, and managing files in a Linux environment.

Execution of these commands-

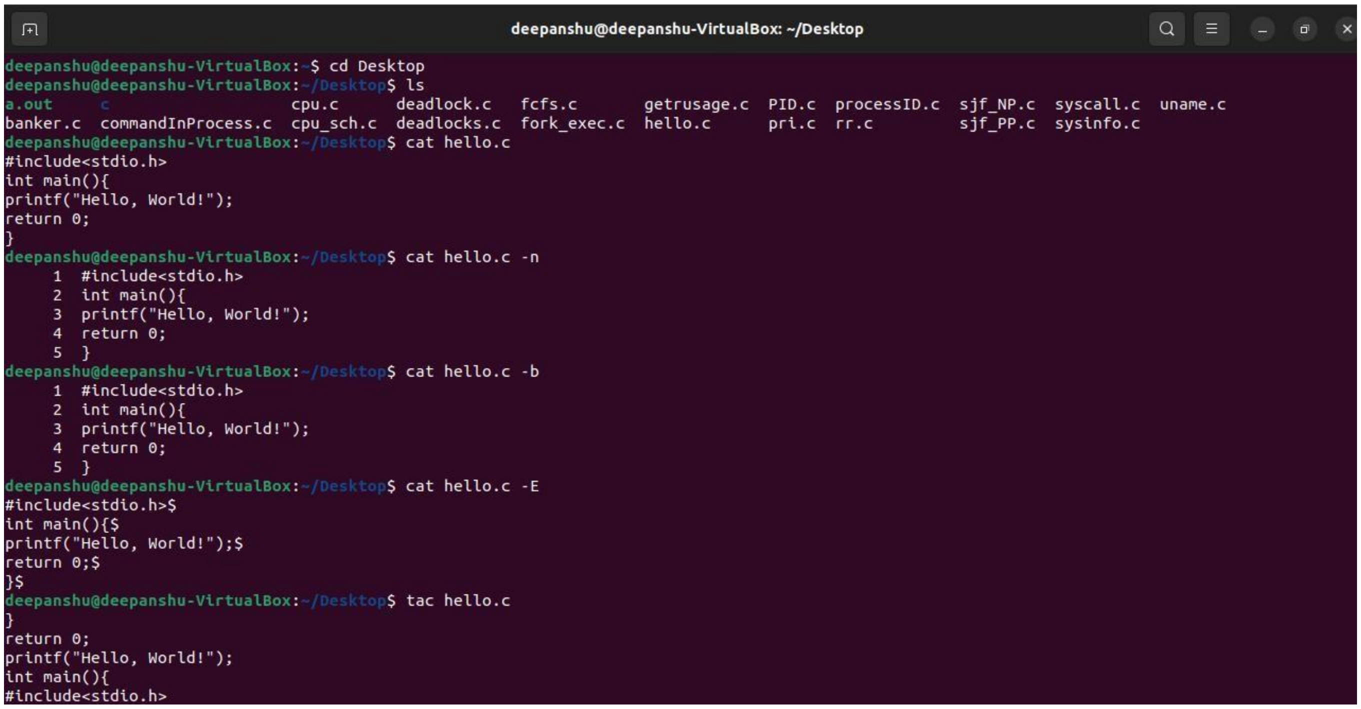
Open the terminal in the linux and these commands will be executed there.



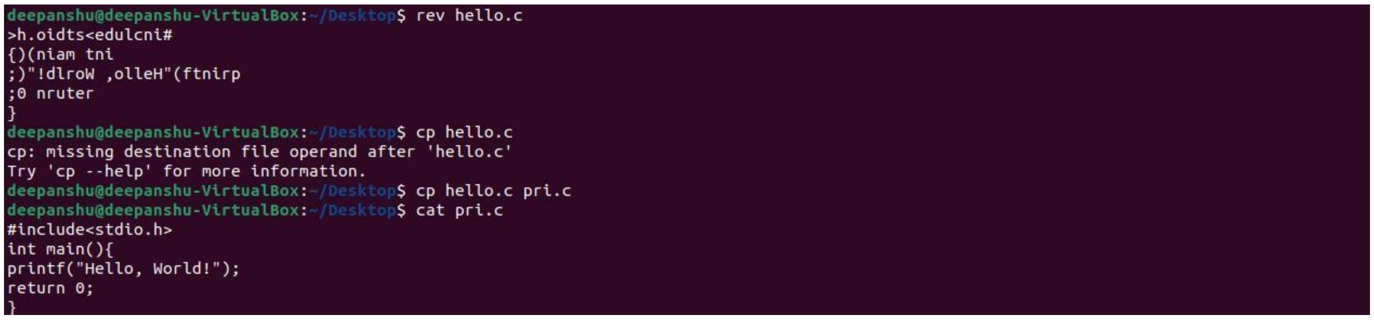
```
deepanshu@deepanshu-VirtualBox: ~/Desktop
deepanshu@deepanshu-VirtualBox:~/Desktop$ which ls
/usr/bin/ls
deepanshu@deepanshu-VirtualBox:~/Desktop$ whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
deepanshu@deepanshu-VirtualBox:~/Desktop$ whatis ls
ls (1)               - list directory contents
```



```
deepanshu@deepanshu-VirtualBox: ~
deepanshu@deepanshu-VirtualBox:~$ ls
a.out  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos  world
deepanshu@deepanshu-VirtualBox:~$ ls -l
total 68
-rwxrwxr-x 1 deepanshu deepanshu 16104 Feb 21 13:35 a.out
drwxr-xr-x 2 deepanshu deepanshu  4096 Apr 18 19:52 Desktop
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Documents
drwxr-xr-x 2 deepanshu deepanshu  4096 Feb  9 09:13 Downloads
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Music
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Pictures
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Public
drwx------ 4 deepanshu deepanshu  4096 Feb  2 09:52 snap
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Templates
drwxr-xr-x 2 deepanshu deepanshu  4096 Jan 12 10:30 Videos
-rwxrwxr-x 1 deepanshu deepanshu 15960 Feb 21 11:36 world
deepanshu@deepanshu-VirtualBox:~$ ls -a
.   a.out          .bash_logout  .cache   Desktop    Downloads  .local  Pictures  Public  .ssh                       Templates  world
..  .bash_history  .bashrc       .config  Documents  .gnupg     Music   .profile  snap    .sudo_as_admin_successful  Videos
deepanshu@deepanshu-VirtualBox:~$ ls -lh
total 68K
-rwxrwxr-x 1 deepanshu deepanshu  16K Feb 21 13:35 a.out
drwxr-xr-x 2 deepanshu deepanshu 4.0K Apr 18 19:52 Desktop
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Documents
drwxr-xr-x 2 deepanshu deepanshu 4.0K Feb  9 09:13 Downloads
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Music
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Pictures
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Public
drwx------ 4 deepanshu deepanshu 4.0K Feb  2 09:52 snap
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Templates
drwxr-xr-x 2 deepanshu deepanshu 4.0K Jan 12 10:30 Videos
-rwxrwxr-x 1 deepanshu deepanshu  16K Feb 21 11:36 world
deepanshu@deepanshu-VirtualBox:~$ ls -t
Desktop  a.out  world  Downloads  snap  Documents  Music  Pictures  Public  Templates  Videos
deepanshu@deepanshu-VirtualBox:~$ ls -r
world  Videos  Templates  snap  Public  Pictures  Music  Downloads  Documents  Desktop  a.out
deepanshu@deepanshu-VirtualBox:~$ ls -s
total 68
16 a.out   4 Desktop   4 Documents   4 Downloads   4 Music   4 Pictures   4 Public   4 snap   4 Templates   4 Videos  16 world
deepanshu@deepanshu-VirtualBox:~$ ls -i
1239126 a.out    1232169 Documents  1232170 Music     1232168 Public  1232167 Templates  1239161 world
1232147 Desktop  1232149 Downloads  1232171 Pictures  1233541 snap    1232172 Videos
deepanshu@deepanshu-VirtualBox:~$ ls -color
total 68
-rwxrwxr-x 1 deepanshu 15960 Feb 21 11:36 world
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Videos
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Templates
drwx------ 4 deepanshu  4096 Apr 18 19:31 snap
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Public
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Pictures
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Music
drwxr-xr-x 2 deepanshu  4096 Feb  9 09:13 Downloads
drwxr-xr-x 2 deepanshu  4096 Jan 12 10:30 Documents
drwxr-xr-x 2 deepanshu  4096 Apr 18 19:52 Desktop
-rwxrwxr-x 1 deepanshu 16104 Feb 21 13:35 a.out
deepanshu@deepanshu-VirtualBox:~$ 
```

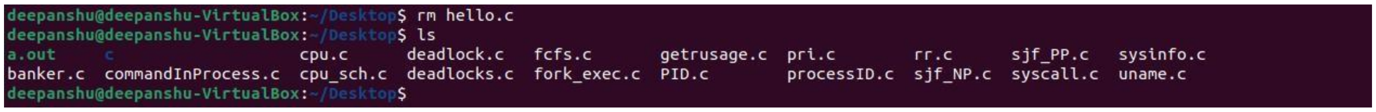


```
deepanshu@deepanshu-VirtualBox:~$ cd Desktop
deepanshu@deepanshu-VirtualBox:~/Desktop$ ls
a.out     c                  cpu.c      deadlock.c   fcfs.c       getrusage.c  PID.c  processID.c  sjf_NP.c  syscall.c  uname.c
banker.c  commandInProcess.c  cpu_sch.c  deadlocks.c  fork_exec.c  hello.c      pri.c  rr.c         sjf_PP.c  sysinfo.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ cat hello.c
#include<stdio.h>
int main(){
printf("Hello, World!");
return 0;
}
deepanshu@deepanshu-VirtualBox:~/Desktop$ cat hello.c -n
     1  #include<stdio.h>
     2  int main(){
     3  printf("Hello, World!");
     4  return 0;
     5  }
deepanshu@deepanshu-VirtualBox:~/Desktop$ cat hello.c -b
     1  #include<stdio.h>
     2  int main(){
     3  printf("Hello, World!");
     4  return 0;
     5  }
deepanshu@deepanshu-VirtualBox:~/Desktop$ cat hello.c -E
#include<stdio.h>$
int main(){$
printf("Hello, World!");$
return 0;$
}$
deepanshu@deepanshu-VirtualBox:~/Desktop$ tac hello.c
}
return 0;
printf("Hello, World!");
int main(){
#include<stdio.h>
```



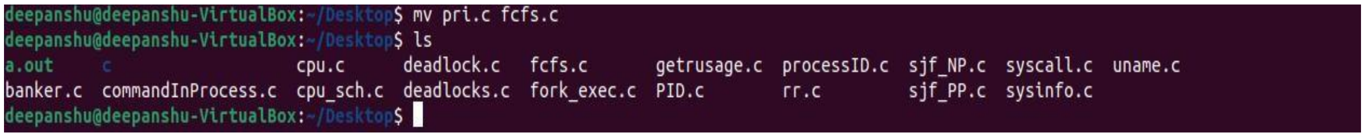
```
deepanshu@deepanshu-VirtualBox:~/Desktop$ rev hello.c
>h.oidts<edulcni#
{)(niam tni
;)"!dlroW ,olleH"(ftnirp
;0 nruter
}
deepanshu@deepanshu-VirtualBox:~/Desktop$ cp hello.c
cp: missing destination file operand after 'hello.c'
Try 'cp --help' for more information.
deepanshu@deepanshu-VirtualBox:~/Desktop$ cp hello.c pri.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ cat pri.c
#include<stdio.h>
int main(){
printf("Hello, World!");
return 0;
}
```

Here, as it can be seen that the data of hello.txt moved to pri.c .



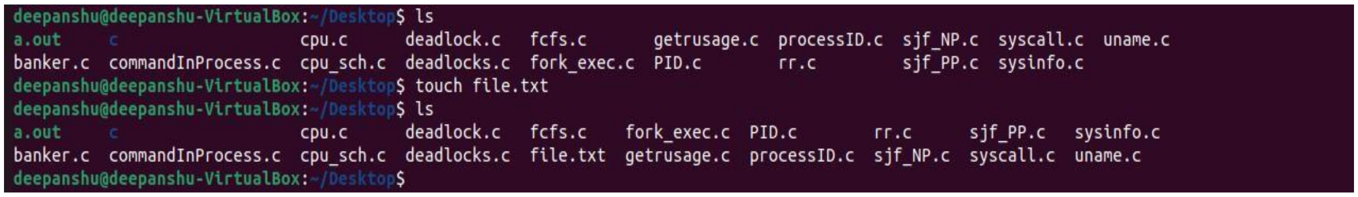
```
deepanshu@deepanshu-VirtualBox:~/Desktop$ rm hello.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ ls
a.out     c                  cpu.c      deadlock.c   fcfs.c       getrusage.c  pri.c        rr.c      sjf_PP.c   sysinfo.c
banker.c  commandInProcess.c  cpu_sch.c  deadlocks.c  fork_exec.c  PID.c        processID.c  sjf_NP.c  syscall.c  uname.c
deepanshu@deepanshu-VirtualBox:~/Desktop$
```

Here, as it can be seen that the hello.c file got removed.



```
deepanshu@deepanshu-VirtualBox:~/Desktop$ mv pri.c fcfs.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ ls
a.out     c                  cpu.c      deadlock.c   fcfs.c       getrusage.c  processID.c  sjf_NP.c  syscall.c  uname.c
banker.c  commandInProcess.c  cpu_sch.c  deadlocks.c  fork_exec.c  PID.c        rr.c         sjf_PP.c  sysinfo.c
deepanshu@deepanshu-VirtualBox:~/Desktop$
```

Here, as it can be seen that the contents of pri.c got moved to fcfs.c .



```
deepanshu@deepanshu-VirtualBox:~/Desktop$ ls
a.out     c                  cpu.c      deadlock.c   fcfs.c       getrusage.c  processID.c  sjf_NP.c  syscall.c  uname.c
banker.c  commandInProcess.c  cpu_sch.c  deadlocks.c  fork_exec.c  PID.c        rr.c         sjf_PP.c  sysinfo.c
deepanshu@deepanshu-VirtualBox:~/Desktop$ touch file.txt
deepanshu@deepanshu-VirtualBox:~/Desktop$ ls
a.out     c                  cpu.c      deadlock.c   fcfs.c    fork_exec.c  PID.c        rr.c      sjf_PP.c   sysinfo.c
banker.c  commandInProcess.c  cpu_sch.c  deadlocks.c  file.txt  getrusage.c  processID.c  sjf_NP.c  syscall.c  uname.c
deepanshu@deepanshu-VirtualBox:~/Desktop$
```

Here, as it can be seen that new file named file.txt got created.