UNIX & SHELL PROGRAMMING LAB MANUAL
(II YEAR – I SEMESTER)

(2019-20)

Prepared by

Sri.M.Naga Raju
Assistant Professor

Ms. L. L. Praneetha
Assistant Professor

GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institution with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru – 521356

GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institution with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru – 521356

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE VISION & MISSION

Institute Vision:

To be a leading institution of engineering education and research, preparing students for leadership in their fields in a caring and challenging learning environment.

Institute Mission:

- To produce quality engineers by providing state-of-the-art engineering education.
- To attract and retain knowledgeable, creative, motivated and highly skilled individuals whose leadership and contributions uphold the college tenets of education, creativity, research and responsible public service.
- To develop faculty and resources to impart and disseminate knowledge and information to students and also to society that will enhance educational level, which in turn, will contribute to social and economic betterment of society.
- To provide an environment that values and encourages knowledge acquisition and academic freedom, making this a preferred institution for knowledge seekers.
- To provide quality assurance.
- To partner and collaborate with industry, government, and R&D institutes to develop new knowledge and sustainable technologies and serve as an engine for facilitating the nation's economic development.
- To impart personality development skills to students that will help them to succeed and lead.
- To instil in students the attitude, values and vision that will prepare them to lead lives of personal integrity and civic responsibility.
- To promote a campus environment that welcomes and makes students of all races, cultures and civilizations feel at home.
- Putting students face to face with industrial, governmental and societal challenges.

DEPARTMENT VISION & MISSION

VISION

To be a centre of innovation by adopting changes in Information Technology, imparting quality education, research to produce visionary computer professionals and entrepreneurs.

MISSION

- To provide an academic environment in which students are given the essential resources for solving real-world problems and work in multidisciplinary teams.
- To impart value based education and research among students, particularly belonging to rural areas, for their sustained growth in technological aspects and leadership.
- To collaborate with the industry for making the students adoptable to evolving changes in Information Technology and related areas.

<u>PROGRAMME EDUCATIONAL OBJECTIVES(PEOs):-</u>

PEO1: To exhibit analytical skills in modeling and solving computing problems by applying mathematical, scientific and engineering knowledge and to pursue their higher studies.

PEO2: To communicate effectively with multi-disciplinary teams to develop quality software systems with an orientation towards research and development for lifelong

PEO3: To address industry and societal needs for the growth of global economy using emerging technologies by following professional ethics.

<u>PROGRAM OUTCOMES (POs)</u>

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

Students will be able to

13.

PSO1: Organize, maintain and protect IT Infrastructural resources.

PSO2: Design and Develop web, mobile, and smart apps based software solutions to the real

CONTENTS

S.No	List of Experiments	Page No
1	To practice on UNIX commands: man, echo, passwd, uname, who, date, cal, banner, tty, pwd, cd, mkdir, rmdir, ls, cp, mv, rm, cat,touch, clear, more, wc, cmp, diff, comm, head, tail, cut, paste, sort, tr, uniq	6
2	To use file permissions	20
3	To Practice process, disk, network utilities	22
4	To practice UNIX commands on Vi Editor	25
5	Write a shell script to print the factorial of first n natural numbers.	28
6	Write a shell script to generate a multiplication table of the given number.	29
7	Write a shell script that counts the number of lines and words present in a given file.	30
8	Write a shell script that displays the list of all files in the given directory.	31
9	Write a shell script (small calculator) that adds, subtracts, multiplies and divides the given two integers.There are two division options: one returns the quotient and the other returns reminder. The script requires 3 arguments: The operation to be used and two integer numbers. The options are add (-a), subtract (-s), multiply (-m), quotient (-c) and reminder (-r).	32
10	Write a program that takes one or more file/directory names as command line input and reports the following information on the file: File type, Number of links, Time of last access, Read, Write and Execute permissions.	34
11	Write a C program that illustrates uses of the opendir, readdir, and closedir APIs.	36
12	Write a C program that counts the number of blanks in a text file using standard I/O	37
13	Implement in C, Unix command 'cat' using system calls.	38
14	Write a C program that illustrates the creation of child process using fork system call.	39
15	Write a C program that illustrates how to execute two commands concurrently with a command pipe.	40
16	Write a C program to implement the Kill function.	41
17	Write a C program to implement the raise function.	42
18	Write a C program that displays the real time of a day every 60 seconds.	43
01	**Additional Programs:** Write a shell script to display files which has read, write and execute permissions.	44-45
02	Write a shell script to display the biggest number in given input numbers.	44-45
03	Write a C program that illustrates the creation of child process using fork system call. One process finds sum of even series and other process finds sum of odd series.	44-45

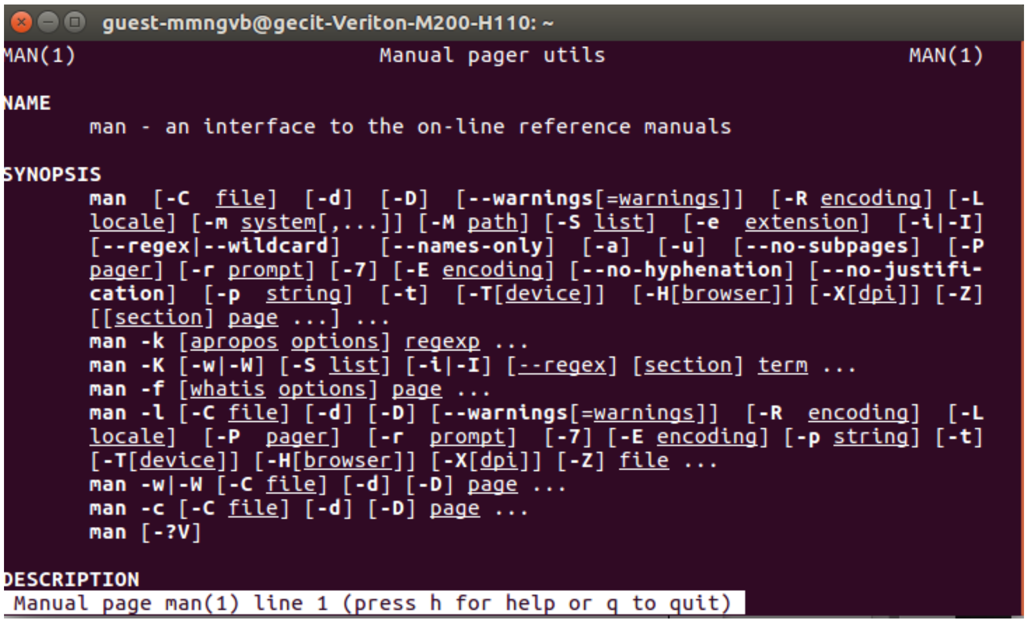
1. To practice on UNIX commands: man, echo, passwd, uname, who, whoami, date, cal, bc, banner, tty, pwd, cd, mkdir, rmdir, ls, cp, mv, rm, cat,touch, clear, wc, cmp, diff, comm, head, tail, cut, paste, sort, tr, uniq

➢ **man:**
man command is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes `NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS`.

Syntax: $man [OPTION]... [COMMAND NAME]…

Input: $man man

Output:



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

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-C  file]  [-d]  [-D]  [--warnings[=warnings]]  [-R encoding] [-L
       locale] [-m system[,...]] [-M path] [-S list]  [-e  extension]  [-i|-I]
       [--regex|--wildcard]   [--names-only]  [-a]  [-u]  [--no-subpages]  [-P
       pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justifi‐
       cation]  [-p  string]  [-t]  [-T[device]]  [-H[browser]] [-X[dpi]] [-Z]
       [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
       man -f [whatis options] page ...
       man -l [-C file] [-d] [-D] [--warnings[=warnings]]  [-R  encoding]  [-L
       locale]  [-P  pager]  [-r  prompt]  [-7] [-E encoding] [-p string] [-t]
       [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
       man -w|-W [-C file] [-d] [-D] page ...
       man -c [-C file] [-d] [-D] page ...
       man [-?V]

DESCRIPTION
 Manual page man(1) line 1 (press h for help or q to quit)
```

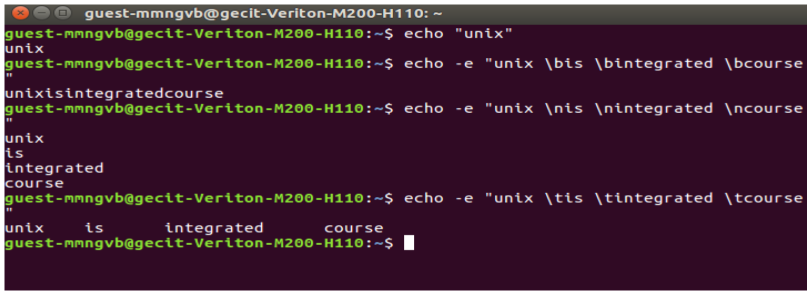
➢ **echo:**
echo command is used to display the line of text/string that are passed as an argument. This is a built-in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.

Syntax: $echo [option] [string]

Options:

1. **\b :** it removes all the spaces in between the text.
2. **\n :** this option creates new line from where it is used.
3. **\t :** this option is used to create horizontal tab spaces.

Output:

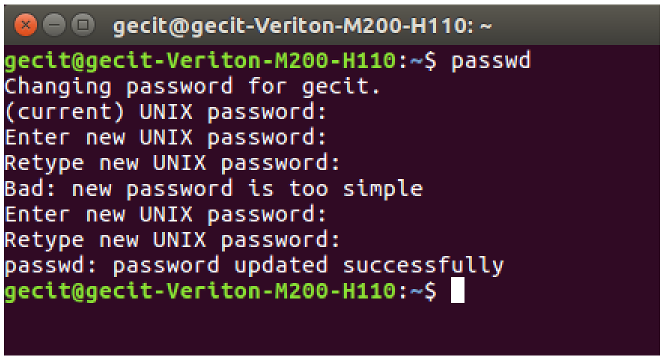


```
guest-mmngvb@gecit-Veriton-M200-H110:~$ echo "unix"
unix
guest-mmngvb@gecit-Veriton-M200-H110:~$ echo -e "unix \bis \bintegrated \bcourse
"
unixisintegratedcourse
guest-mmngvb@gecit-Veriton-M200-H110:~$ echo -e "unix \nis \nintegrated \ncourse
"
unix
is
integrated
course
guest-mmngvb@gecit-Veriton-M200-H110:~$ echo -e "unix \tis \tintegrated \tcourse
"
unix    is      integrated      course
guest-mmngvb@gecit-Veriton-M200-H110:~$
```

➢ **passwd:**

passwd command is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax: $passwd [options] [username]



```
gecit@gecit-Veriton-M200-H110:~$ passwd
Changing password for gecit.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Bad: new password is too simple
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
gecit@gecit-Veriton-M200-H110:~$
```

➢ **uname:**

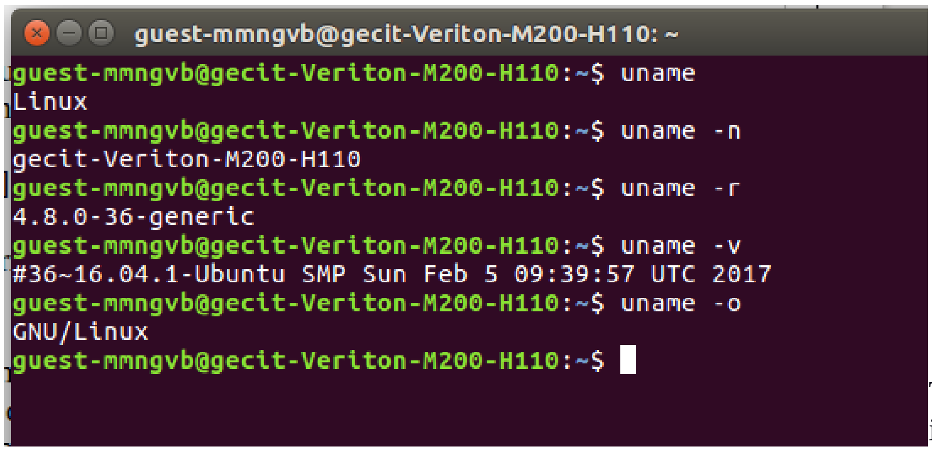
The command ‘uname‘ displays the information about the system.

Syntax: $uname [OPTION]

Options:

1. **-n option**: It prints the hostname of the network node(current computer).
2. **-v option**: It prints the version of the current kernel.
3. **-o option**: It prints the name of the operating system.

Output:



```
guest-mmngvb@gecit-Veriton-M200-H110:~$ uname
Linux
guest-mmngvb@gecit-Veriton-M200-H110:~$ uname -n
gecit-Veriton-M200-H110
guest-mmngvb@gecit-Veriton-M200-H110:~$ uname -r
4.8.0-36-generic
guest-mmngvb@gecit-Veriton-M200-H110:~$ uname -v
#36~16.04.1-Ubuntu SMP Sun Feb 5 09:39:57 UTC 2017
guest-mmngvb@gecit-Veriton-M200-H110:~$ uname -o
GNU/Linux
guest-mmngvb@gecit-Veriton-M200-H110:~$
```

➢ **who:**

The who command is used to get

information about currently logged in user on to system.

Syntax : $who [options] [filename]

The who command displays the following information for each user currently logged in to the system if no option is provided :

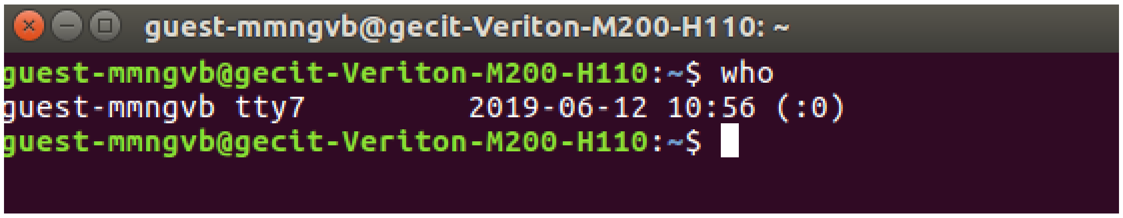
1.Login name of the users

2.Terminal line numbers

3.Login time of the users in to system

4.Remote host name of the user

Output:



```
guest-mmngvb@gecit-Veriton-M200-H110:~$ who
guest-mmngvb tty7         2019-06-12 10:56 (:0)
guest-mmngvb@gecit-Veriton-M200-H110:~$ 
```

- **date:**

date command is used to display the system date and time.

Syntax: date [OPTION]... [+FORMAT]

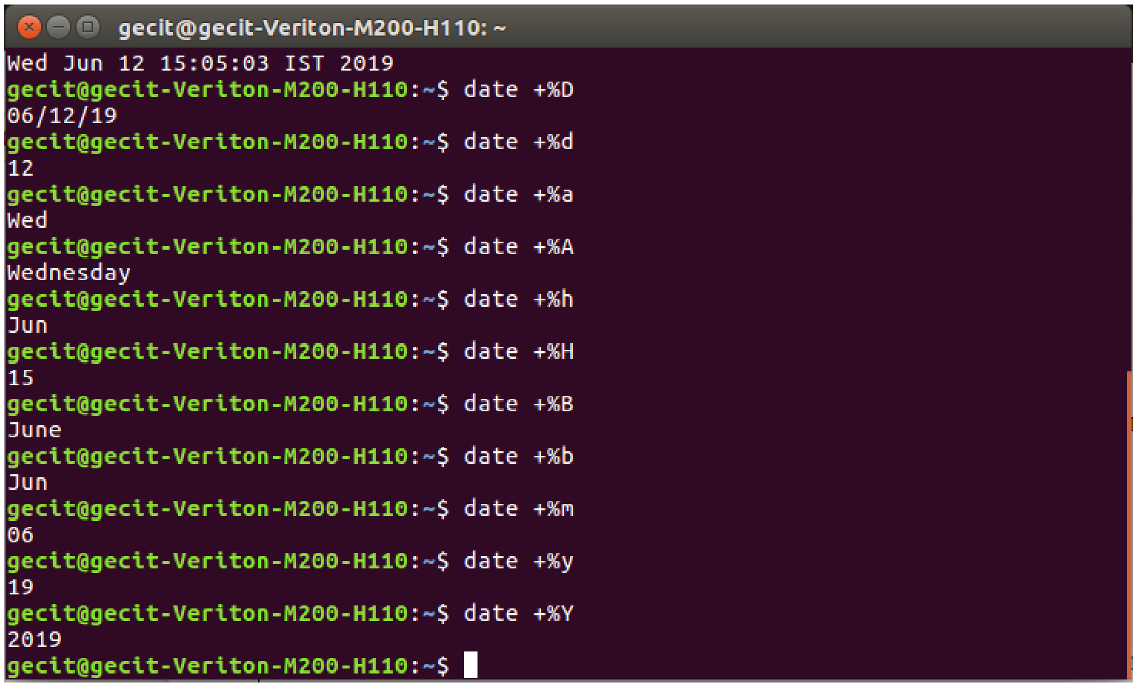
Options:

1. date (no option) : With no options, the date command displays the current date and time, including the abbreviated day name, abbreviated month name, day of the month, the time separated by colons, the time zone name, and the year.

2. List of Format specifiers used with date command:

%D: Display date as mm/dd/yy.
%d: Display the day of the month (01 to 31).
%a: Displays the abbreviated name for weekdays (Sun to Sat).
%A: Displays full weekdays (Sunday to Saturday).
%h: Displays abbreviated month name (Jan to Dec).
%b: Displays abbreviated month name (Jan to Dec).
%B: Displays full month name(January to December).
%m: Displays the month of year (01 to 12).
%y: Displays last two digits of the year(00 to 99).
%Y: Display four-digit year.
%T: Display the time in 24 hour format as HH:MM:SS.
%H: Display the hour.
%M: Display the minute.
%S: Display the seconds.

OUTPUT:



```
gecit@gecit-Veriton-M200-H110: ~
Wed Jun 12 15:05:03 IST 2019
gecit@gecit-Veriton-M200-H110:~$ date +%D
06/12/19
gecit@gecit-Veriton-M200-H110:~$ date +%d
12
gecit@gecit-Veriton-M200-H110:~$ date +%a
Wed
gecit@gecit-Veriton-M200-H110:~$ date +%A
Wednesday
gecit@gecit-Veriton-M200-H110:~$ date +%h
Jun
gecit@gecit-Veriton-M200-H110:~$ date +%H
15
gecit@gecit-Veriton-M200-H110:~$ date +%B
June
gecit@gecit-Veriton-M200-H110:~$ date +%b
Jun
gecit@gecit-Veriton-M200-H110:~$ date +%m
06
gecit@gecit-Veriton-M200-H110:~$ date +%y
19
gecit@gecit-Veriton-M200-H110:~$ date +%Y
2019
gecit@gecit-Veriton-M200-H110:~$ 
```

- **cal:**
  `cal` command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

  Syntax: `cal [ [ month ] year]`

Options:

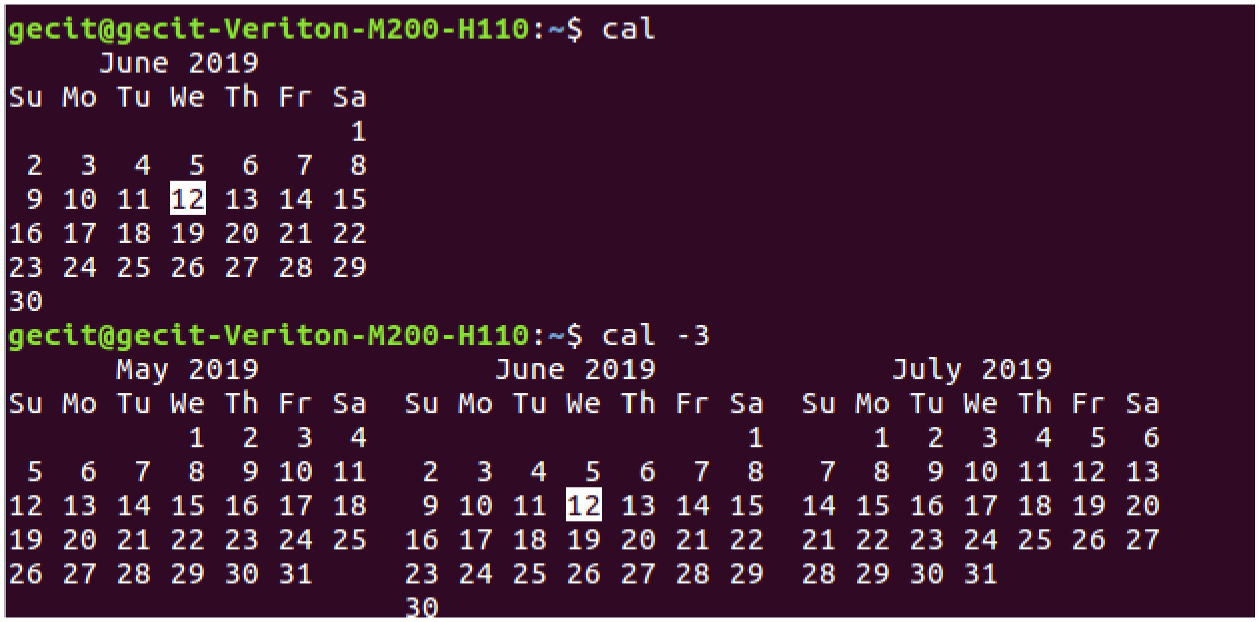
1.**cal :** Shows current month calendar on the terminal.

2.cal 08 2000 : Shows calendar of selected month and year.

3. **cal 2019 :** Shows the whole calendar of the year.

4. **cal -3 :** Shows calendar of previous, current and next month

Output:



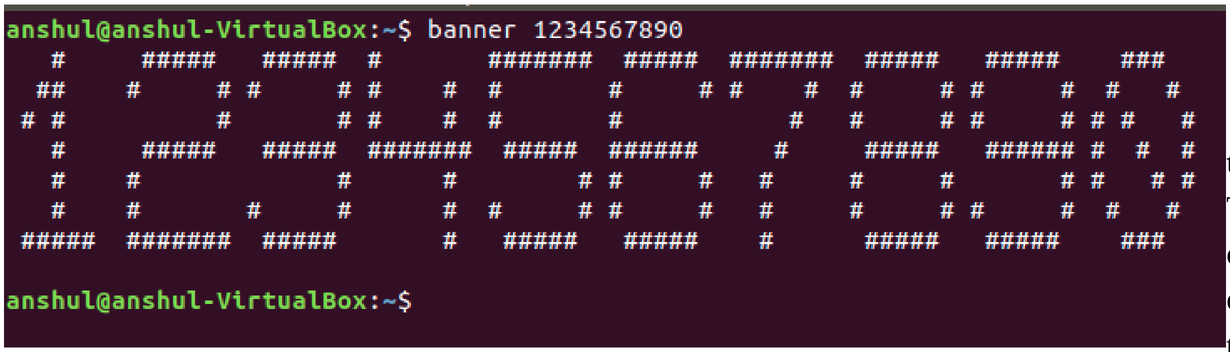
```
gecit@gecit-Veriton-M200-H110:~$ cal
     June 2019
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
gecit@gecit-Veriton-M200-H110:~$ cal -3
      May 2019             June 2019             July 2019
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                     1      1  2  3  4  5  6
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                      30
```

- **banner:**

banner command is used to print the ASCII character string in large letter to standrad output.

Syntax: banner text

Output: $banner 1234567890



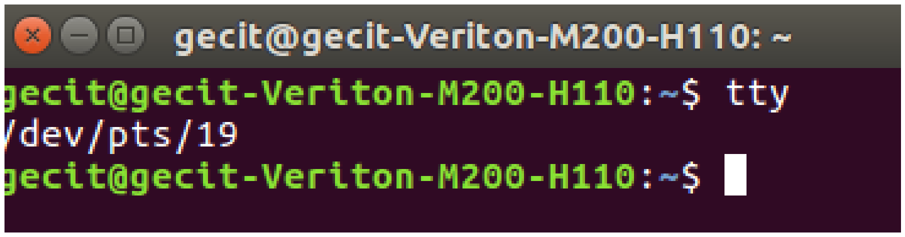
```
anshul@anshul-VirtualBox:~$ banner 1234567890
   #     #####   #####  #       #######  #####  #######  #####   #####    ###
  ##    #     #       # #    #  #       #     # #    #  #     # #     #  #   #
 # #          #       # #    #  #       #           #    #     # #     # # #   #
   #     #####   #####  #######  ######  ######     #     #####   ###### #  #  #
   #    #             #      #         # #     #   #     #     #       # #   # #
   #    #       #     #      #   #     # #     #   #     #     # #     #  #   #
 #####  #######  #####       #    #####   #####    #      #####   #####    ###

anshul@anshul-VirtualBox:~$
```

- **tty:**

The **tty**command of terminal basically prints the file name of the terminal connected to standard input. **tty** is short of teletype, but popularly known as a terminal.

Syntax: tty [OPTION]....

Output:

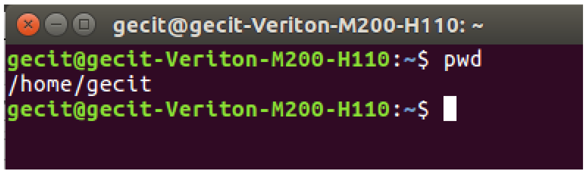


```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ tty
/dev/pts/19
gecit@gecit-Veriton-M200-H110:~$
```

- **pwd:**

pwd stands for **P**rint **W**orking **D**irectory. It prints the path of the working directory, starting from the root. It is a shell built-in command(pwd).

Syntax: tty [OPTION]....

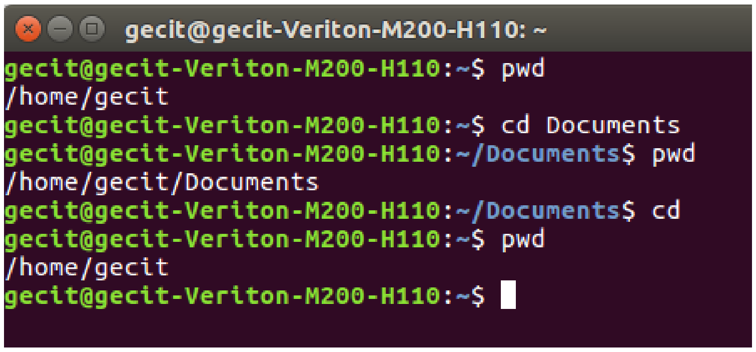


```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ pwd
/home/gecit
gecit@gecit-Veriton-M200-H110:~$
```

- **cd:**

cd command is known as change directory command. It is used to change current working directory.

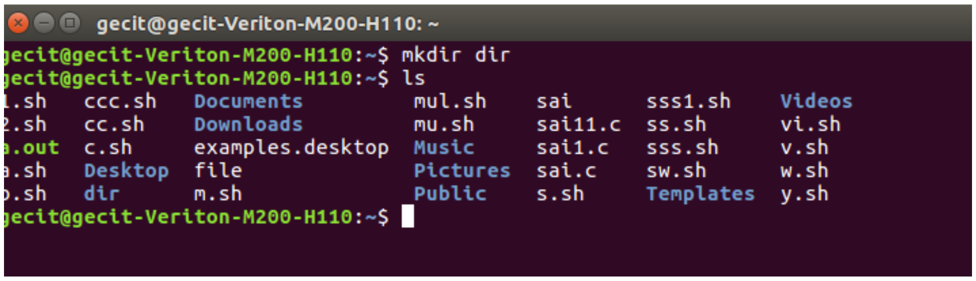
Syntax: $cd [directory]



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ pwd
/home/gecit
gecit@gecit-Veriton-M200-H110:~$ cd Documents
gecit@gecit-Veriton-M200-H110:~/Documents$ pwd
/home/gecit/Documents
gecit@gecit-Veriton-M200-H110:~/Documents$ cd
gecit@gecit-Veriton-M200-H110:~$ pwd
/home/gecit
gecit@gecit-Veriton-M200-H110:~$ 
```

- **mkdir:**
  mkdir command allows the user to create directories.This command can create multiple directories at once.

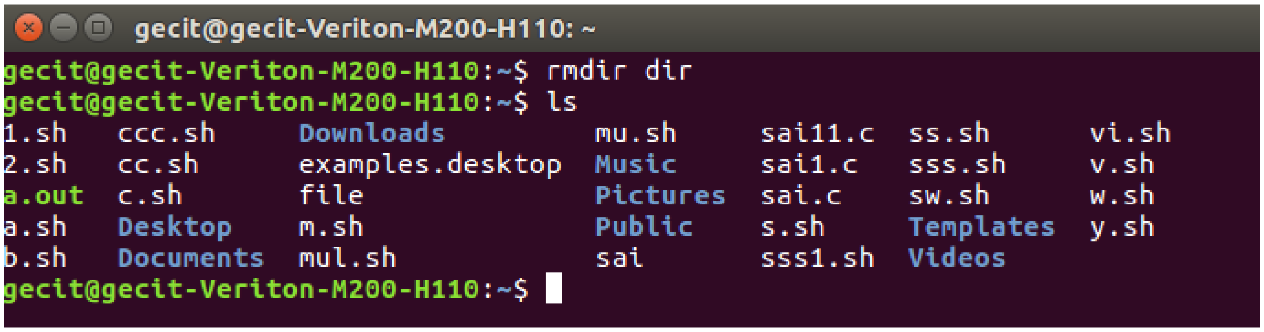
  Syntax: mkdir [options...] [directories …]



```
gecit@gecit-Veriton-M200-H110: ~
ecit@gecit-Veriton-M200-H110:~$ mkdir dir
ecit@gecit-Veriton-M200-H110:~$ ls
.sh    ccc.sh   Documents          mul.sh    sai       sss1.sh    Videos
.sh    cc.sh    Downloads          mu.sh     sai11.c   ss.sh      vi.sh
.out   c.sh     examples.desktop   Music     sai1.c    sss.sh     v.sh
.sh    Desktop  file               Pictures  sai.c     sw.sh      w.sh
.sh    dir      m.sh               Public    s.sh      Templates  y.sh
ecit@gecit-Veriton-M200-H110:~$ 
```

- **rmdir**:
  rmdir command is used remove empty directories from the filesystem.

  Syntax: rmdir [options...] [directories ...]



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ rmdir dir
gecit@gecit-Veriton-M200-H110:~$ ls
1.sh    ccc.sh     Downloads          mu.sh      sai11.c  ss.sh      vi.sh
2.sh    cc.sh      examples.desktop   Music      sai1.c   sss.sh     v.sh
a.out   c.sh       file               Pictures   sai.c    sw.sh      w.sh
a.sh    Desktop    m.sh               Public     s.sh     Templates  y.sh
b.sh    Documents  mul.sh             sai        sss1.sh  Videos
gecit@gecit-Veriton-M200-H110:~$ 
```

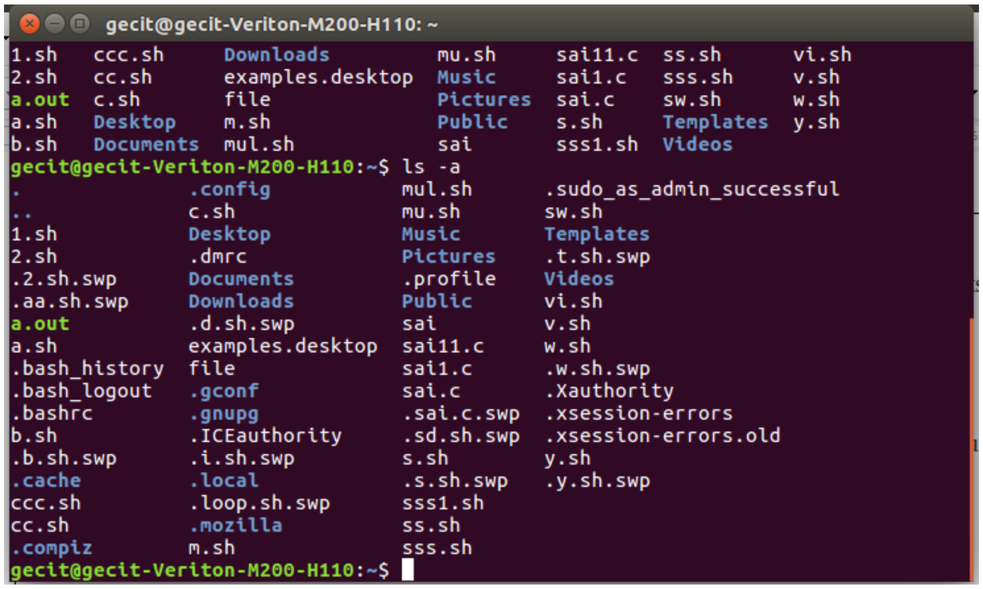
- **ls:**
  ls is the most widely used command in unix. ls command is used to list the contents of a directory.

  Syntax: ls [options] [pathnames]

Options:

1. **ls –a:**To display the hidden files and directories in the current directory.

2. ls –l: The -l option provides lots of information about the file type, owner, group, permissions, file size, last modification date.
3. **ls –x:**The -x option specifies the ls command to display the files in columns.



```
gecit@gecit-Veriton-M200-H110: ~
1.sh    ccc.sh     Downloads          mu.sh     sai11.c  ss.sh      vi.sh
2.sh    cc.sh      examples.desktop   Music     sai1.c   sss.sh     v.sh
a.out   c.sh       file               Pictures  sai.c    sw.sh      w.sh
a.sh    Desktop    m.sh               Public    s.sh     Templates  y.sh
b.sh    Documents  mul.sh             sai       sss1.sh  Videos
gecit@gecit-Veriton-M200-H110:~$ ls -a
.               .config           mul.sh      .sudo_as_admin_successful
..              c.sh              mu.sh       sw.sh
1.sh            Desktop           Music       Templates
2.sh            .dmrc             Pictures    .t.sh.swp
.2.sh.swp       Documents         .profile    Videos
.aa.sh.swp      Downloads         Public      vi.sh
a.out           .d.sh.swp         sai         v.sh
a.sh            examples.desktop  sai11.c     w.sh
.bash_history   file              sai1.c      .w.sh.swp
.bash_logout    .gconf            sai.c       .Xauthority
.bashrc         .gnupg            .sai.c.swp  .xsession-errors
b.sh            .ICEauthority     .sd.sh.swp  .xsession-errors.old
.b.sh.swp       .i.sh.swp         s.sh        y.sh
.cache          .local            .s.sh.swp   .y.sh.swp
ccc.sh          .loop.sh.swp      sss1.sh
cc.sh           .mozilla          ss.sh
.compiz         m.sh              sss.sh
gecit@gecit-Veriton-M200-H110:~$
```

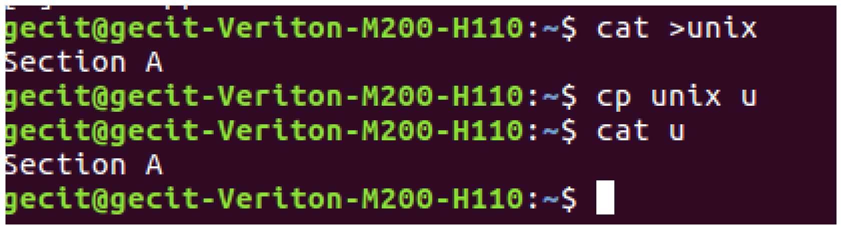
➢ **cp:**

cp stands for **copy**. This command is used to copy files or group of files or directory. *cp* command require at least two filenames in its arguments.

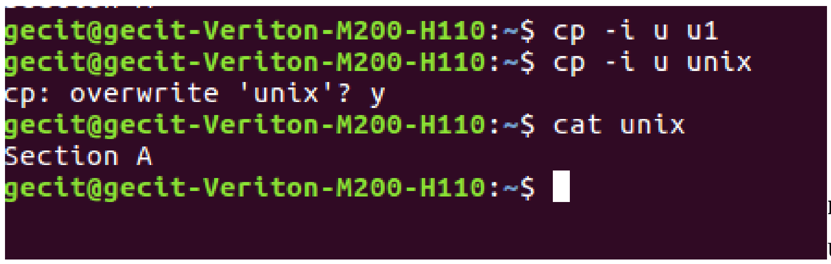
Syntax:cp Src_file Dest_file

Options:

1. **cp -i:i** stands for Interactive copying. With this option system first warns the user before overwriting the destination file. **cp** prompts for a response, if you press **y** then it overwrites the file and with any other option leave it uncopied.
2. **cp –r:** With this option **cp** command shows its recursive behavior by copying the entire directory structure recursively.



```
gecit@gecit-Veriton-M200-H110:~$ cat >unix
Section A
gecit@gecit-Veriton-M200-H110:~$ cp unix u
gecit@gecit-Veriton-M200-H110:~$ cat u
Section A
gecit@gecit-Veriton-M200-H110:~$
```

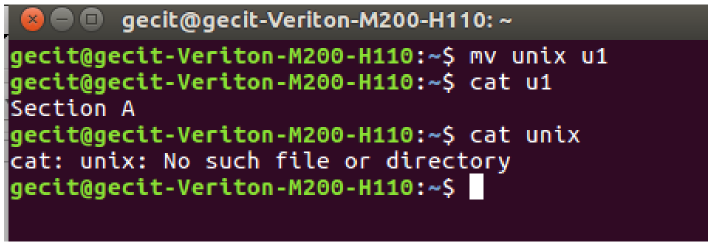


```
gecit@gecit-Veriton-M200-H110:~$ cp -i u u1
gecit@gecit-Veriton-M200-H110:~$ cp -i u unix
cp: overwrite 'unix'? y
gecit@gecit-Veriton-M200-H110:~$ cat unix
Section A
gecit@gecit-Veriton-M200-H110:~$
```

➢ **mv:**

mv stands for **move**. mv is used to move one or more files or directories from one place to another in file system.

Syntax:mv [Option] source destination



```
gecit@gecit-Veriton-M200-H110:~$ mv unix u1
gecit@gecit-Veriton-M200-H110:~$ cat u1
Section A
gecit@gecit-Veriton-M200-H110:~$ cat unix
cat: unix: No such file or directory
gecit@gecit-Veriton-M200-H110:~$
```

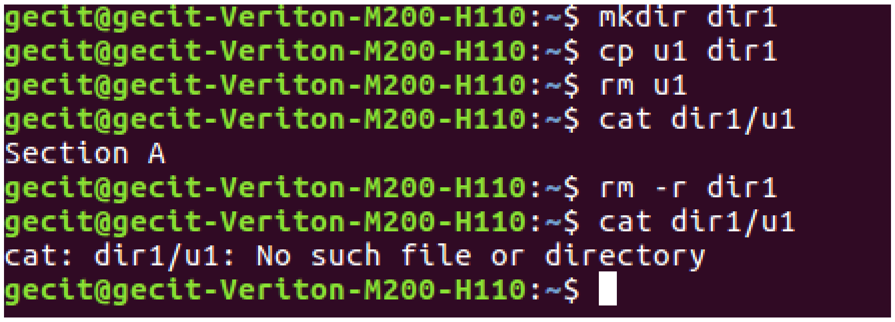
- **rm:**

rm command is used to remove objects such as files, directories, symbolic links and so on from the file system

Syntax: rm [OPTION]... FILE...

Options:

1.rm –r: rm will delete all the files and sub-directories recursively of the parent directory. At each stage it deletes everything it finds. Normally, **rm** wouldn't delete the directories but when used with this option, it will delete.



```
gecit@gecit-Veriton-M200-H110:~$ mkdir dir1
gecit@gecit-Veriton-M200-H110:~$ cp u1 dir1
gecit@gecit-Veriton-M200-H110:~$ rm u1
gecit@gecit-Veriton-M200-H110:~$ cat dir1/u1
Section A
gecit@gecit-Veriton-M200-H110:~$ rm -r dir1
gecit@gecit-Veriton-M200-H110:~$ cat dir1/u1
cat: dir1/u1: No such file or directory
gecit@gecit-Veriton-M200-H110:~$
```

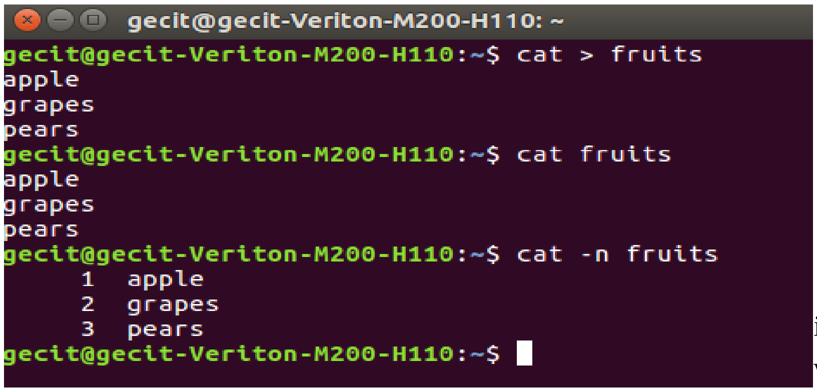
- **cat:**

cat(concatenate) command reads data from the file and gives their content as output. It can create, view, concatenate files.

Syntax: cat [OPTION] [FILE]...

Options:

1. cat –n: Toview contents of a file preceding with line numbers.



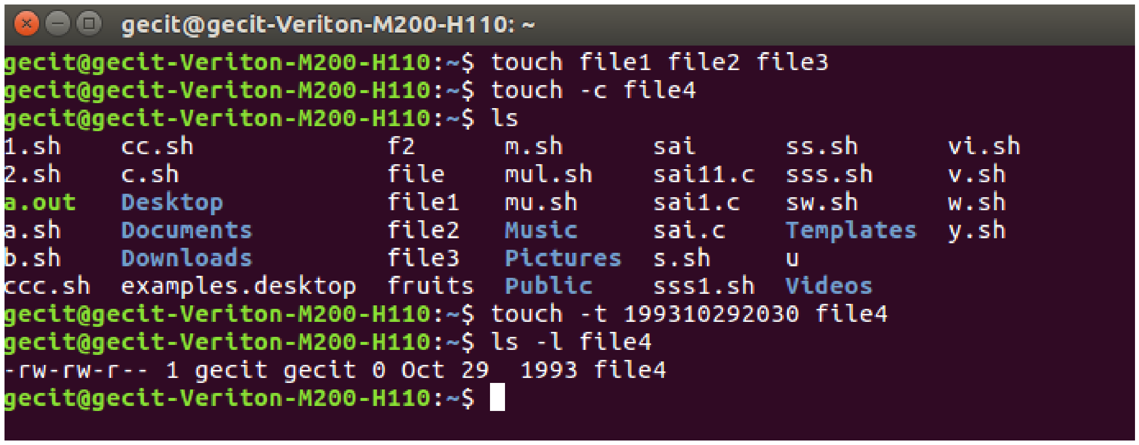
```
gecit@gecit-Veriton-M200-H110:~$ cat > fruits
apple
grapes
pears
gecit@gecit-Veriton-M200-H110:~$ cat fruits
apple
grapes
pears
gecit@gecit-Veriton-M200-H110:~$ cat -n fruits
     1  apple
     2  grapes
     3  pears
gecit@gecit-Veriton-M200-H110:~$
```

- **touch**: It is used to create a file without any content. The file created using touch command is empty.

Syntax: touch File1_name

Options:

1. **touch –c:**Using -c option with touch command avoids creating new files.
2. **touch –m:** It will update the last modification time of the file.
3. **touch -t YYMMDDHHMM.SS:** creates a file with specified time other than the current time.



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ touch file1 file2 file3
gecit@gecit-Veriton-M200-H110:~$ touch -c file4
gecit@gecit-Veriton-M200-H110:~$ ls
1.sh    cc.sh            f2      m.sh      sai      ss.sh      vi.sh
2.sh    c.sh             file    mul.sh    sai11.c  sss.sh     v.sh
a.out   Desktop          file1   mu.sh     sai1.c   sw.sh      w.sh
a.sh    Documents        file2   Music     sai.c    Templates  y.sh
b.sh    Downloads        file3   Pictures  s.sh     u
ccc.sh  examples.desktop fruits  Public    sss1.sh  Videos
gecit@gecit-Veriton-M200-H110:~$ touch -t 199310292030 file4
gecit@gecit-Veriton-M200-H110:~$ ls -l file4
-rw-rw-r-- 1 gecit gecit 0 Oct 29  1993 file4
gecit@gecit-Veriton-M200-H110:~$ 
```

➢ **clear:**Itis a standard Unix computer operating system command that is used to clear the terminal screen.

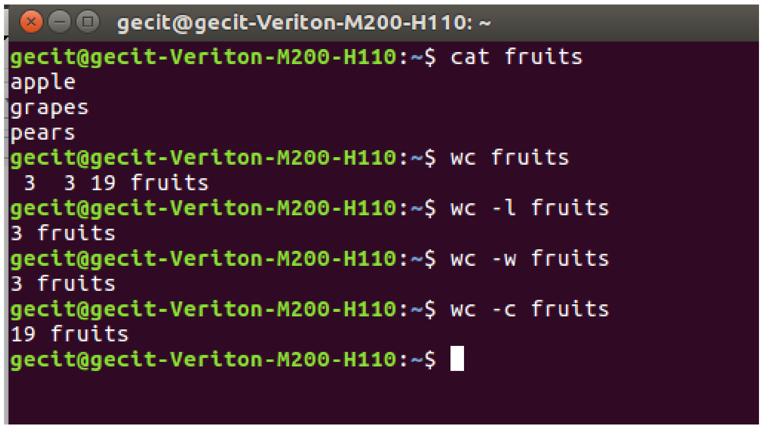
➢ **wc:**

wc stands for word count, it is used to find out number of lines**,** word count**,** byte and characters count in the files .

Syntax: wc File_name

Options:

1. wc -l : Prints the number of lines in a file.
2. wc -w : prints the number of words in a file.
3. wc -c : Displays the count of bytes in a file.



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ cat fruits
apple
grapes
pears
gecit@gecit-Veriton-M200-H110:~$ wc fruits
 3  3 19 fruits
gecit@gecit-Veriton-M200-H110:~$ wc -l fruits
3 fruits
gecit@gecit-Veriton-M200-H110:~$ wc -w fruits
3 fruits
gecit@gecit-Veriton-M200-H110:~$ wc -c fruits
19 fruits
gecit@gecit-Veriton-M200-H110:~$ 
```

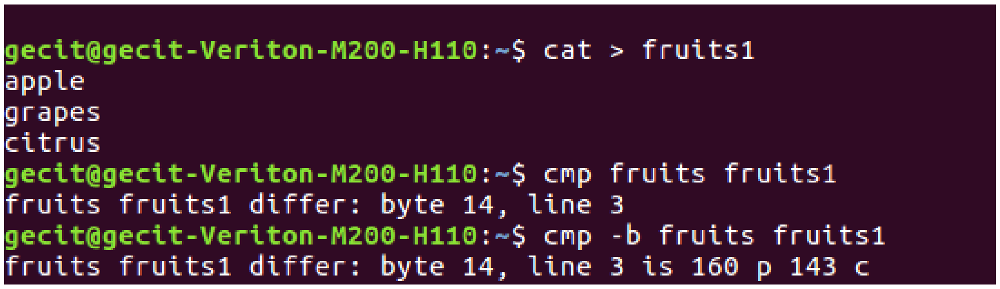
➢ **cmp:** This command is used to compare two files byte by byte.

Syntax:cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

the optional SKIP1 and SKIP2 specify the number of bytes to skip at the beginning of each file which is zero by default and OPTION refers to the options compatible with this command.

Options:

1. **cmp -b(print-bytes) :** cmp displays the differing bytes in the output when used with **-b** option.
2. **cmp -i [bytes-to-be-skipped]** : Now, this option when used with cmp command helps to skipa particular number of initial bytes from both the files and then after skipping it compares the files.



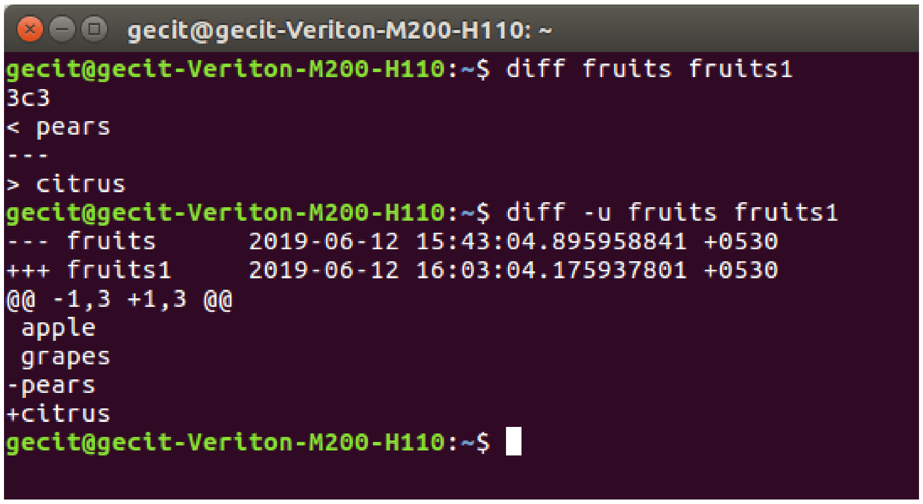
```
gecit@gecit-Veriton-M200-H110:~$ cat > fruits1
apple
grapes
citrus
gecit@gecit-Veriton-M200-H110:~$ cmp fruits fruits1
fruits fruits1 differ: byte 14, line 3
gecit@gecit-Veriton-M200-H110:~$ cmp -b fruits fruits1
fruits fruits1 differ: byte 14, line 3 is 160 p 143 c
```

- **diff:**It stands for difference**.** This command is used to display the differences in the files by comparing the files line by line.

  Syntax:diff [options] File1 File2

Options:

1. **diff -c (context) :** To view differences in context mode, use the **-c** option.
2. **diff -u (unified) :** To view differences in unified mode, use the **-u** option. It is similar to context mode but it doesn't display any redundant information or it shows the information in concise form.



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ diff fruits fruits1
3c3
< pears
---
> citrus
gecit@gecit-Veriton-M200-H110:~$ diff -u fruits fruits1
--- fruits      2019-06-12 15:43:04.895958841 +0530
+++ fruits1     2019-06-12 16:03:04.175937801 +0530
@@ -1,3 +1,3 @@
 apple
 grapes
-pears
+citrus
gecit@gecit-Veriton-M200-H110:~$
```

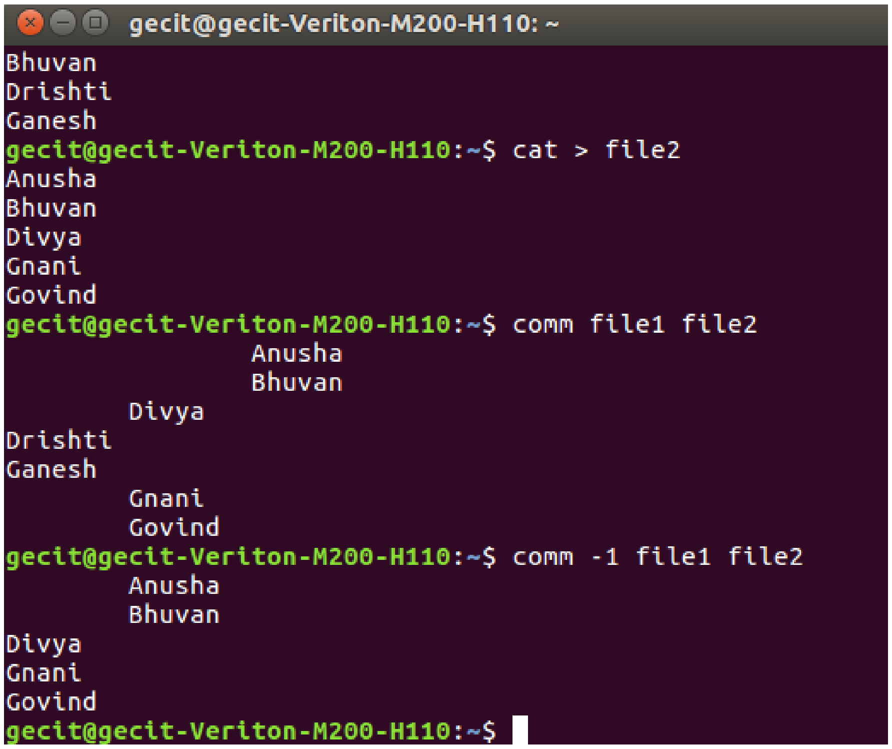
- **comm:** It compare two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.

  Syntax:comm [OPTION]... FILE1 FILE2

Options:

-1 : suppress first column(lines unique to first file).

-2 : suppress second column(lines unique to second file).



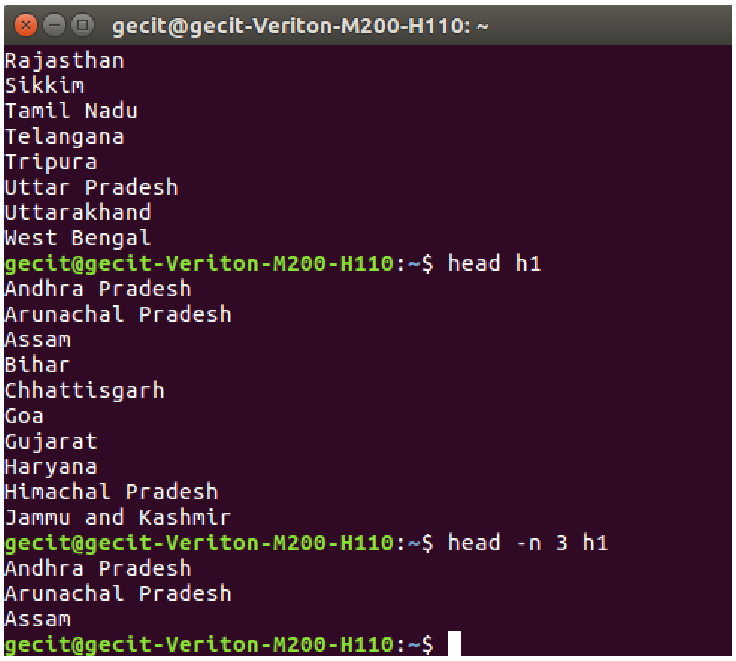
```
gecit@gecit-Veriton-M200-H110: ~
Bhuvan
Drishti
Ganesh
gecit@gecit-Veriton-M200-H110:~$ cat > file2
Anusha
Bhuvan
Divya
Gnani
Govind
gecit@gecit-Veriton-M200-H110:~$ comm file1 file2
                Anusha
                Bhuvan
        Divya
Drishti
Ganesh
        Gnani
        Govind
gecit@gecit-Veriton-M200-H110:~$ comm -1 file1 file2
        Anusha
        Bhuvan
Divya
Gnani
Govind
gecit@gecit-Veriton-M200-H110:~$ 
```

➢ **head:** prints the top N number of data of the given input.

Syntax: head [OPTION]... [FILE]...

Options:

1. head **-**n num**:** Prints the first 'num' lines instead of first 10 lines.
2. head **-**q**:** It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.



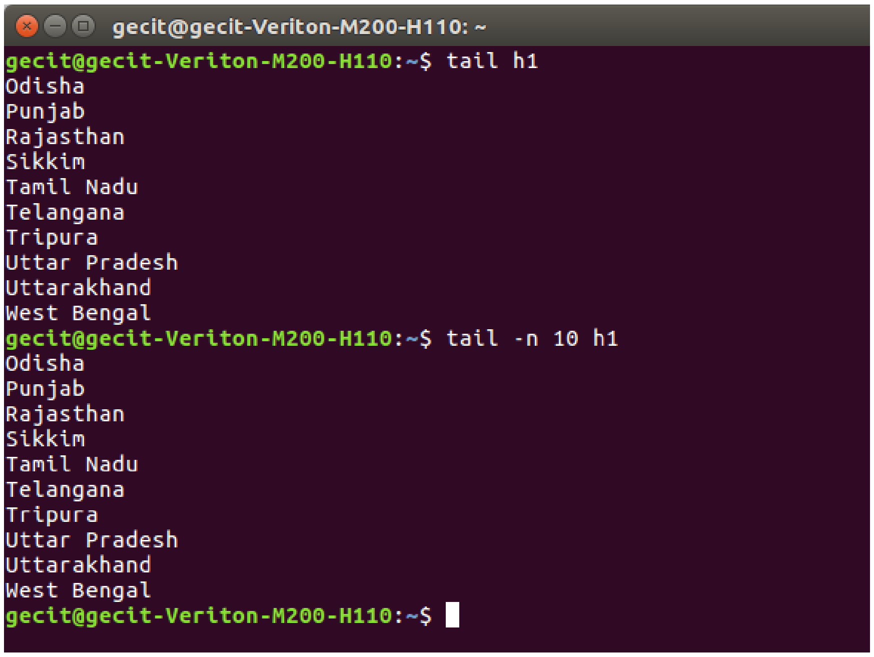
```
gecit@gecit-Veriton-M200-H110: ~
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
gecit@gecit-Veriton-M200-H110:~$ head h1
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
gecit@gecit-Veriton-M200-H110:~$ head -n 3 h1
Andhra Pradesh
Arunachal Pradesh
Assam
gecit@gecit-Veriton-M200-H110:~$ 
```

➢ **tail:**prints the last N number of data of the given input.

Syntax: tail [OPTION]... [FILE]...

Options:

1. tail -n num: Prints the last 'num' lines instead of last 10 lines.
2. tail -q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.



```
gecit@gecit-Veriton-M200-H110:~$ tail h1
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
gecit@gecit-Veriton-M200-H110:~$ tail -n 10 h1
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
gecit@gecit-Veriton-M200-H110:~$
```

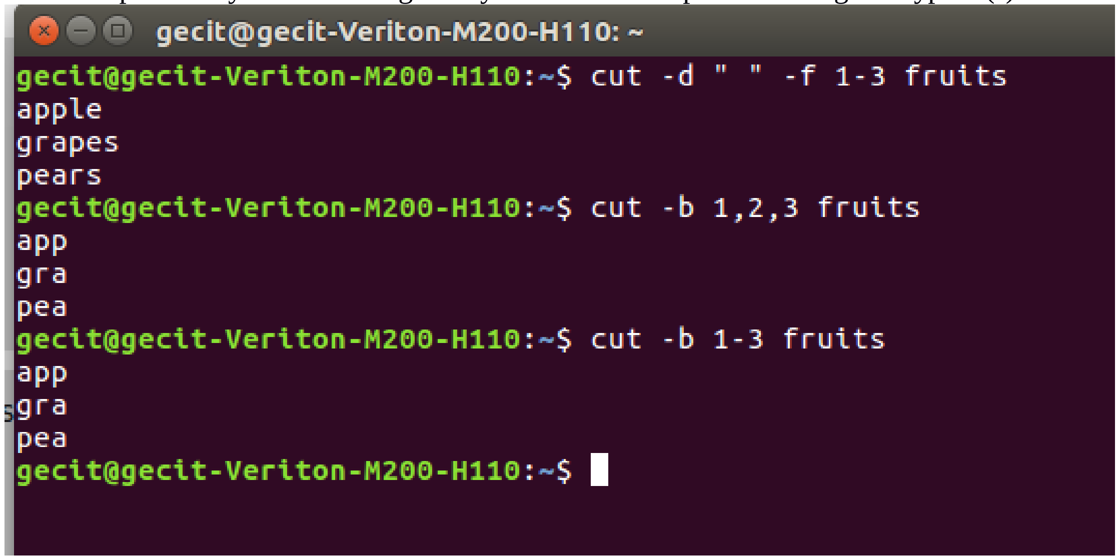
- **cut:**It can be used to cut parts of a line by byte position, character and field**.**

  Syntax:cut OPTION... [FILE]...

Options:

cut -f (field): To extract the useful information you need to cut by fields rather than columns. List of the fields number specified must be separated by comma. Ranges are not described with -f option.

cut -b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-).



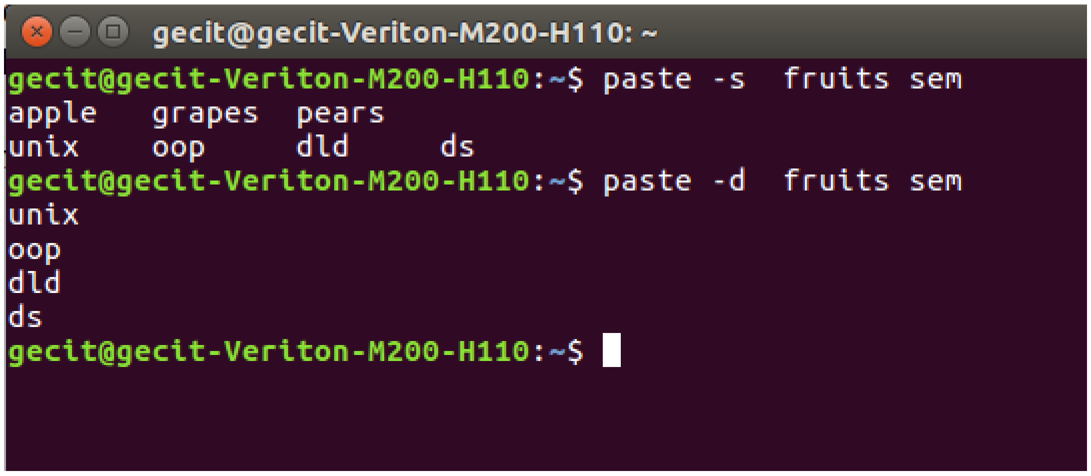
```
gecit@gecit-Veriton-M200-H110:~$ cut -d " " -f 1-3 fruits
apple
grapes
pears
gecit@gecit-Veriton-M200-H110:~$ cut -b 1,2,3 fruits
app
gra
pea
gecit@gecit-Veriton-M200-H110:~$ cut -b 1-3 fruits
app
gra
pea
gecit@gecit-Veriton-M200-H110:~$
```

- **paste:**It is used to join files horizontally by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output.

  Syntax:paste [OPTION]... [FILES]...

Options:

1. **paste -s (serial):** We can merge the files in sequentially manner using the -s option. It reads all the lines from a single file and merges all these lines into a single line with each line separated by tab.
2. **paste -d (delimiter):** Paste command uses the tab delimiter by default for merging the files.



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ paste -s  fruits sem
apple   grapes  pears
unix    oop     dld     ds
gecit@gecit-Veriton-M200-H110:~$ paste -d  fruits sem
unix
oop
dld
ds
gecit@gecit-Veriton-M200-H110:~$ 
```

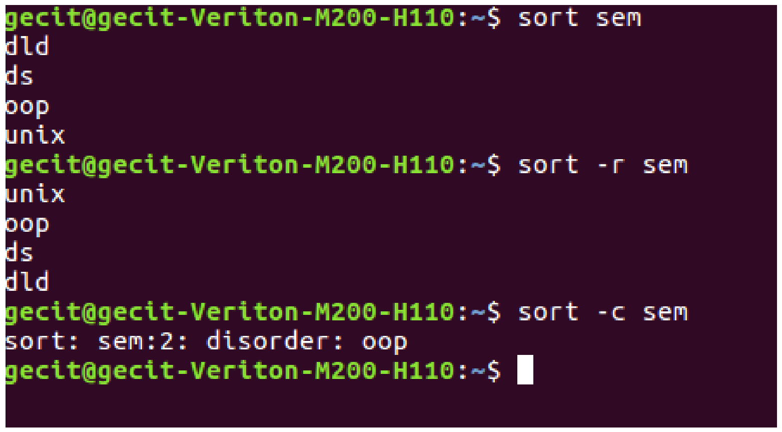
➢ **sort:** sorts the contents of a text file, line by line.

Syntax: sort [options] [files]

Options:

1. sort –r: sorts the input file in reverse order i.e. descending order by default.

2.sort -c: This option is used to check if the file given is already sorted or not.



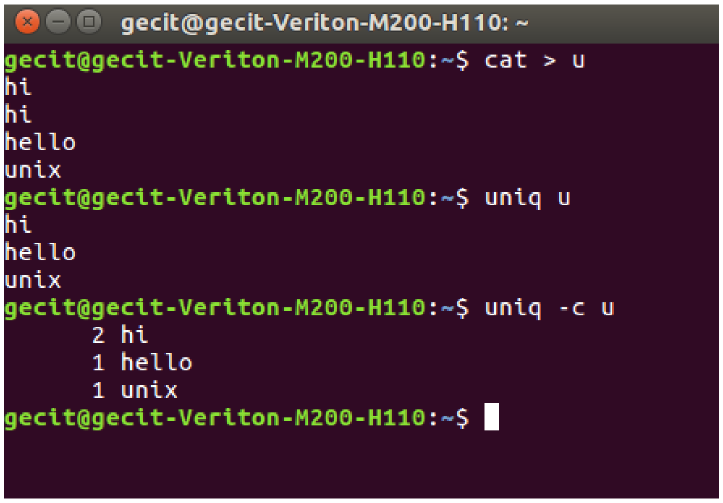
```
gecit@gecit-Veriton-M200-H110:~$ sort sem
dld
ds
oop
unix
gecit@gecit-Veriton-M200-H110:~$ sort -r sem
unix
oop
ds
dld
gecit@gecit-Veriton-M200-H110:~$ sort -c sem
sort: sem:2: disorder: oop
gecit@gecit-Veriton-M200-H110:~$ 
```

➢ **uniq:** It is a command line utility that reports or filters out the repeated lines in a file.

Syntax: uniq [OPTION] [INPUT[OUTPUT]]

Options:

uniq -c: It tells how many times a line was repeated by displaying a number as a prefix with the line.



gecit@gecit-Veriton-M200-H110: ~

```
gecit@gecit-Veriton-M200-H110:~$ cat > u
hi
hi
hello
unix
gecit@gecit-Veriton-M200-H110:~$ uniq u
hi
hello
unix
gecit@gecit-Veriton-M200-H110:~$ uniq -c u
      2 hi
      1 hello
      1 unix
gecit@gecit-Veriton-M200-H110:~$ 
```

2. To use file permissions.

Aim: To implement file permissions

Description:

File Permissions:

Every file and directory in your UNIX system has following 3 permissions defined for the user, group and others.

- **Read:**This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- **Write:**The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
- **Execute:**In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

  '**chmod**' command which stands for 'change mode' can set permissions (read, write, execute) on a file/directory for the owner, group and the world.

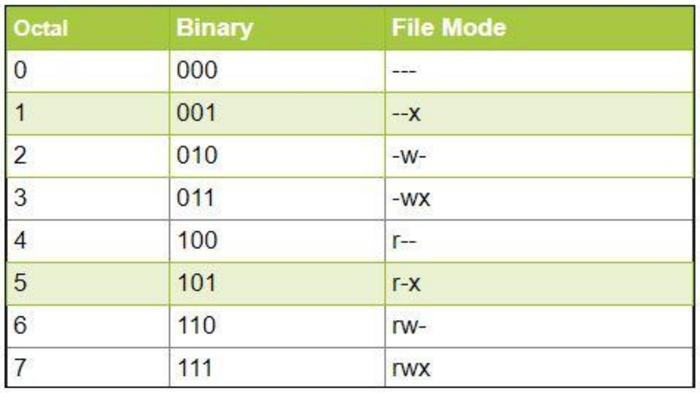
Syntax:chmod permissions filename

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

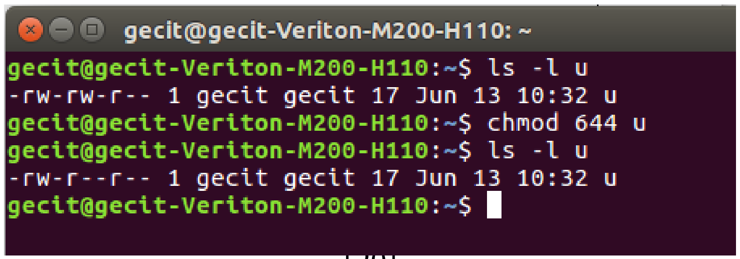
Absolute(Numeric) Mode

In this mode, file permissions are not represented as characters but a three-digit octal number.



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Output:



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ ls -l u
-rw-rw-r-- 1 gecit gecit 17 Jun 13 10:32 u
gecit@gecit-Veriton-M200-H110:~$ chmod 644 u
gecit@gecit-Veriton-M200-H110:~$ ls -l u
-rw-r--r-- 1 gecit gecit 17 Jun 13 10:32 u
gecit@gecit-Veriton-M200-H110:~$
```

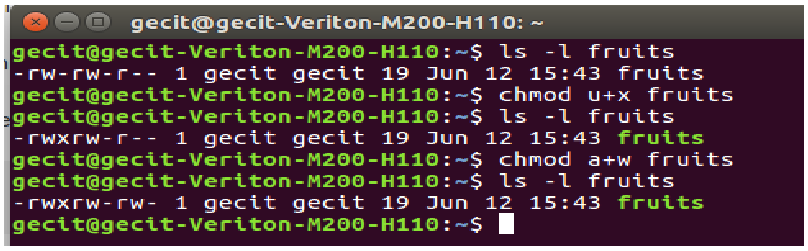
Symbolic Mode: In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

The various owners are represented as -

User Denotations	
u	user/owner
g	group
o	other
a	all

Output:



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ ls -l fruits
-rw-rw-r-- 1 gecit gecit 19 Jun 12 15:43 fruits
gecit@gecit-Veriton-M200-H110:~$ chmod u+x fruits
gecit@gecit-Veriton-M200-H110:~$ ls -l fruits
-rwxrw-r-- 1 gecit gecit 19 Jun 12 15:43 fruits
gecit@gecit-Veriton-M200-H110:~$ chmod a+w fruits
gecit@gecit-Veriton-M200-H110:~$ ls -l fruits
-rwxrw-rw- 1 gecit gecit 19 Jun 12 15:43 fruits
gecit@gecit-Veriton-M200-H110:~$
```

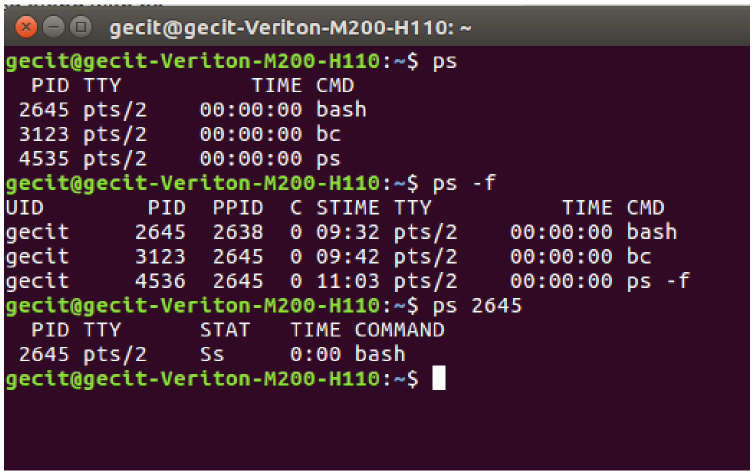
3. To Practice on process, disk, network utilities

Process utilities:

An instance of a program is called a Process. In simple terms, any command that you give to your unix machine starts a new process.

ps:

- ps (Process status) can be used to see/list all the running processes.
- For more information -f (full) can be used along with ps
- For a single process information, ps along with process id is used

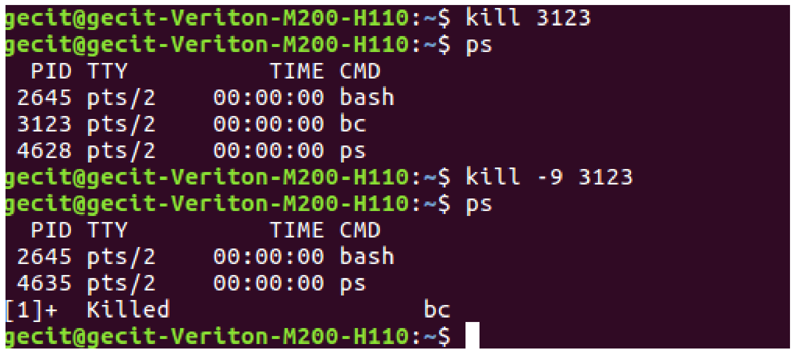


```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ ps
  PID TTY          TIME CMD
 2645 pts/2    00:00:00 bash
 3123 pts/2    00:00:00 bc
 4535 pts/2    00:00:00 ps
gecit@gecit-Veriton-M200-H110:~$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
gecit     2645  2638  0 09:32 pts/2    00:00:00 bash
gecit     3123  2645  0 09:42 pts/2    00:00:00 bc
gecit     4536  2645  0 11:03 pts/2    00:00:00 ps -f
gecit@gecit-Veriton-M200-H110:~$ ps 2645
  PID TTY      STAT   TIME COMMAND
 2645 pts/2    Ss     0:00 bash
gecit@gecit-Veriton-M200-H110:~$ 
```

kill:

When invoked kill command sends a termination signal to the process being killed. We can employ sure kill signal to forcibly terminate a process. Signal number for sure kill is 9

$ kill Pid:



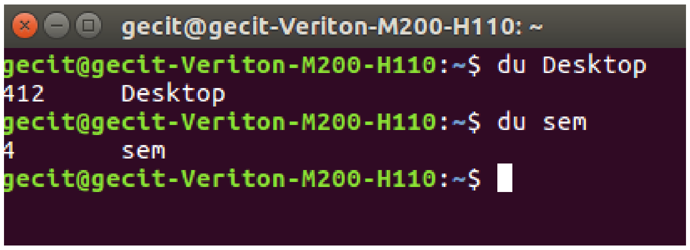
```
gecit@gecit-Veriton-M200-H110:~$ kill 3123
gecit@gecit-Veriton-M200-H110:~$ ps
  PID TTY          TIME CMD
 2645 pts/2    00:00:00 bash
 3123 pts/2    00:00:00 bc
 4628 pts/2    00:00:00 ps
gecit@gecit-Veriton-M200-H110:~$ kill -9 3123
gecit@gecit-Veriton-M200-H110:~$ ps
  PID TTY          TIME CMD
 2645 pts/2    00:00:00 bash
 4635 pts/2    00:00:00 ps
[1]+  Killed                  bc
gecit@gecit-Veriton-M200-H110:~$ 
```

Disk Utilities:

du:(Disk usuage)

du command-line utility helps you to find out the disk usage of set of files or a directory.

Syntax: du [OPTION]... [FILE]…

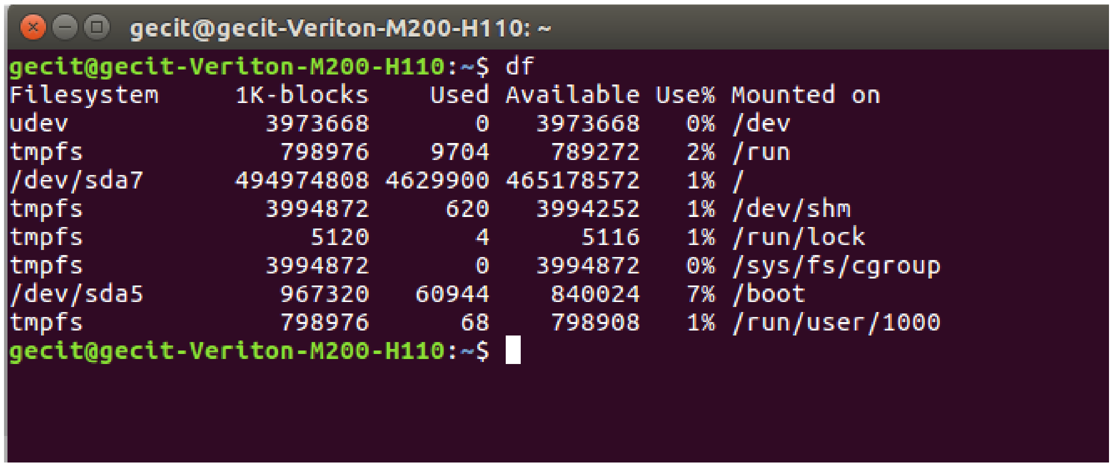


```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ du Desktop
412     Desktop
gecit@gecit-Veriton-M200-H110:~$ du sem
4       sem
gecit@gecit-Veriton-M200-H110:~$ 
```

df:(Disk freespace)

df command displays the amount of disk space available on the file system containing each file name argument.

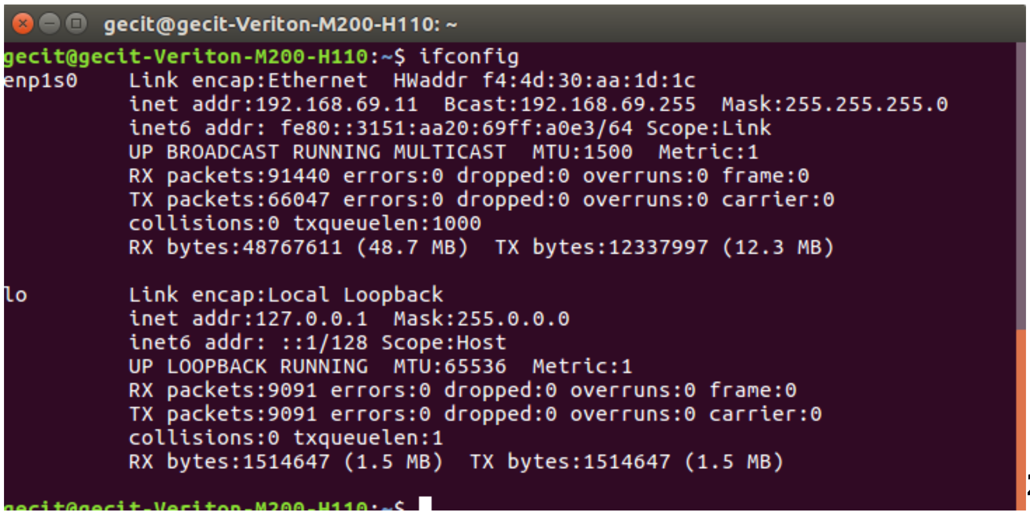
Syntax:df [OPTION]...[FILE]...



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev             3973668       0   3973668   0% /dev
tmpfs             798976    9704    789272   2% /run
/dev/sda7      494974808 4629900 465178572   1% /
tmpfs            3994872     620   3994252   1% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
tmpfs            3994872       0   3994872   0% /sys/fs/cgroup
/dev/sda5         967320   60944    840024   7% /boot
tmpfs             798976      68    798908   1% /run/user/1000
gecit@gecit-Veriton-M200-H110:~$ 
```

Network Utilities:

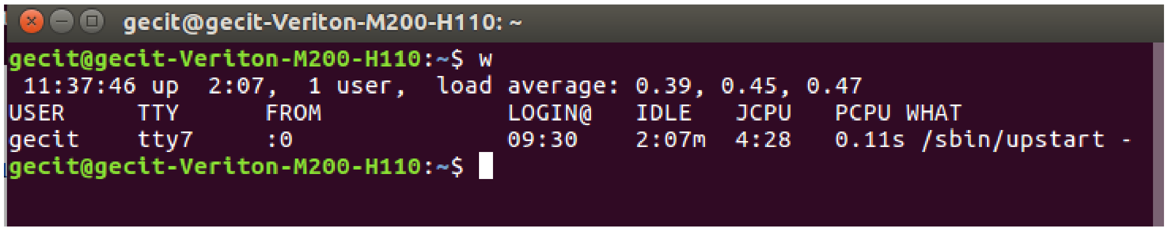
1.ifconfig: It is used to configure network interface parameters.



```
gecit@gecit-Veriton-M200-H110: ~
gecit@gecit-Veriton-M200-H110:~$ ifconfig
enp1s0    Link encap:Ethernet  HWaddr f4:4d:30:aa:1d:1c
          inet addr:192.168.69.11  Bcast:192.168.69.255  Mask:255.255.255.0
          inet6 addr: fe80::3151:aa20:69ff:a0e3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:91440 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66047 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48767611 (48.7 MB)  TX bytes:12337997 (12.3 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:9091 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9091 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1514647 (1.5 MB)  TX bytes:1514647 (1.5 MB)

gecit@gecit-Veriton-M200-H110:~$ 
```

2.w:

w prints a summary of the current activity on the system, including what each user is doing, and their processes.



gecit@gecit-Veriton-M200-H110: ~

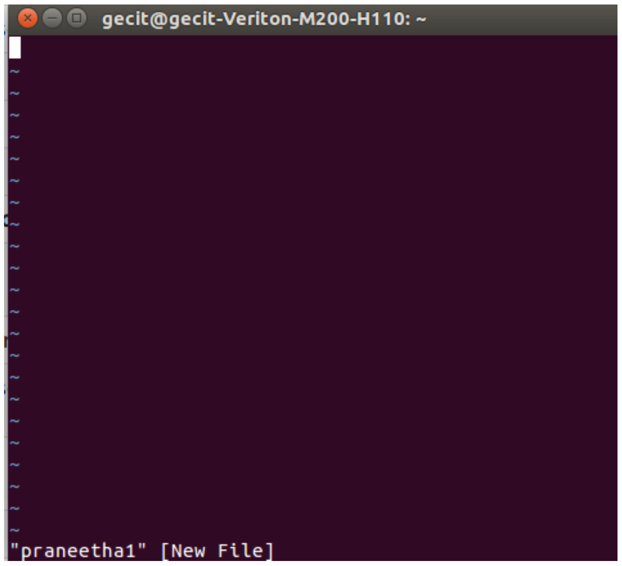
```
gecit@gecit-Veriton-M200-H110:~$ w
 11:37:46 up  2:07,  1 user,  load average: 0.39, 0.45, 0.47
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
gecit    tty7     :0               09:30    2:07m  4:28   0.11s /sbin/upstart -
gecit@gecit-Veriton-M200-H110:~$ 
```

4. To practice UNIX commands on Vi Editor.

Vi Editor: The default editor that comes with the UNIX operating system is called vi (visual editor). Using vi editor, we can edit an existing file or create a new file. We can also use this editor to just read a text file.

Syntax: vi filename

$vi praneetha1



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"praneetha1" [New File]
```

To work on VI editor, you need to understand its operation modes. They can be divided into two main parts.

1.Command mode:
The vi editor opens in this mode, and it only understands commands

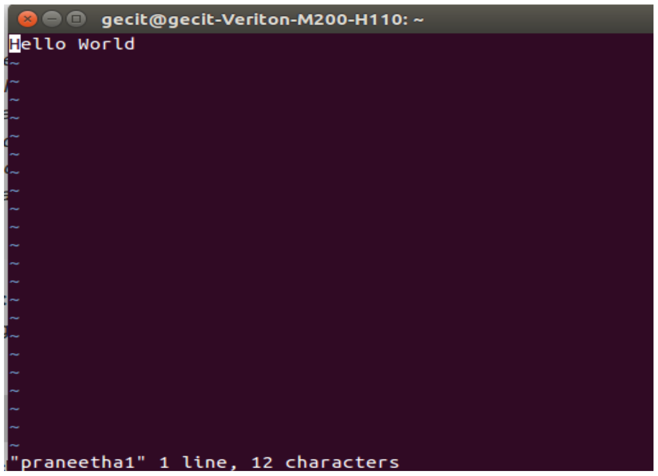
- In this mode, you can, move the cursor and cut, copy, paste the text
- This mode also saves the changes you have made to the file
- Commands are case sensitive. You should use the right letter case.

2. Insert mode:

- This mode is for inserting text in the file.
- You can switch to the Insert mode from the command mode by pressing 'i' on the keyboard
- Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
- To return to the command mode and save the changes you have made you need to press the Esc key

vi Editing commands:

Note: You should be in the "command mode" to execute these commands. VI editor is case- sensitive so make sure you type the commands in the right letter-case.



```
Hello World
~
~
"praneetha1" 1 line, 12 characters
```

To Exit vi:

	`:x<Return>`	*quit `vi`, writing out modified file to file named in original invocation*
	`:wq<Return>`	*quit `vi`, writing out modified file to file named in original invocation*
	`:q<Return>`	*quit (or exit) `vi`*

Inserting or Adding Text:

	i	*Insert text before cursor, until <Esc> hit*
	I	*insert text at beginning of current line, until <Esc> hit*
	a	*append text after cursor, until <Esc> hit*
	A	*append text to end of current line, until <Esc> hit*
	o	*open and put text in a new line below current line, until <Esc> hit*
	O	*open and put text in a new line above current line, until <Esc> hit*

Changing Text:

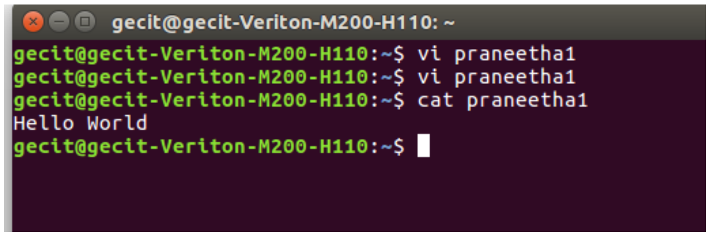
	r	*replace single character under cursor (no <Esc> needed)*
	R	*replace characters, starting with current cursor position, until <Esc> hit*
	cw	*change the current word with new text,* *starting with the character under cursor, until <Esc> hit*
	C	*change (replace) the characters in the current line, until <Esc> hit*
	cc	*change (replace) the entire current line, stopping when <Esc> is hit*

Deleting Text:

	x	*delete single character under cursor*

	dw	*delete the single word beginning with character under cursor*
	D	*delete the remainder of the line, starting with current cursor position*
	dd	*ire current line*

Output:



```
gecit@gecit-Veriton-M200-H110:~$ vi praneetha1
gecit@gecit-Veriton-M200-H110:~$ vi praneetha1
gecit@gecit-Veriton-M200-H110:~$ cat praneetha1
Hello World
gecit@gecit-Veriton-M200-H110:~$ 
```

Practice remaining commands in experiment1 on vi editor

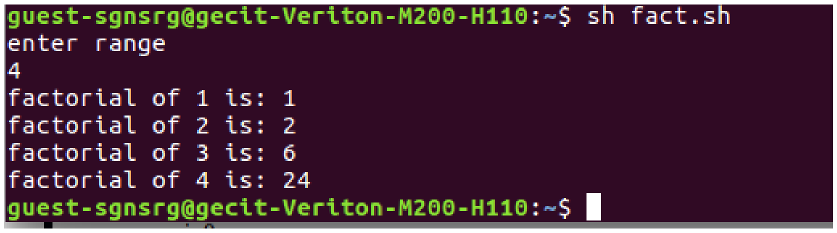
5. Write a shell script to print the factorial of first n natural numbers.

Aim: To print the factorial of first n natural numbers.

Program:

```
echo "enter range"
read n
i=1
while [ $i -le $n ]
do
j=1 fact=1
while [ $j -le $i ]
do
fact=`expr $fact \* $j`
j=`expr $j + 1`
done
echo factorial of $i is: $fact
i=`expr $i + 1`
done
```

Output:



```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ sh fact.sh
enter range
4
factorial of 1 is: 1
factorial of 2 is: 2
factorial of 3 is: 6
factorial of 4 is: 24
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

6. Write a shell script to generate a multiplication table of the given number.

Aim: To generate a multiplication table of the given number.

Description:

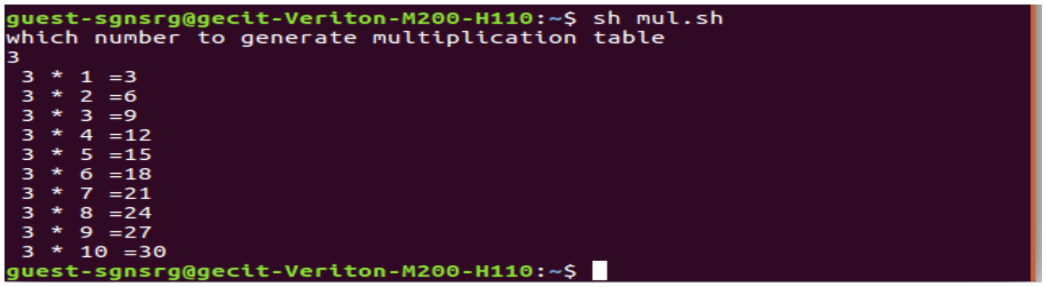
To display any information on the screen we can use ‘echo ‘ command . The text within the double quotes was displayed on the screen. ‘expr’ command is used to evaluate the expression. The expression within the symbol ‘ ‘ is evaluated by using the expr command $variable gives the value of the variable it contained expr command combines two functions perfumes arithmetic operations on integers manipulate strings

The operand +,-, * etc.., must be enclosed on either side by white space.

Program:

```
echo "which number to generate multiplication table"
read number
i=1
while [ $i -le 10 ]
do
echo " $number * $i =`expr $number \* $i ` "
i=`expr $i + 1`
done
```

Output:



```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ sh mul.sh
which number to generate multiplication table
3
 3 * 1 =3
 3 * 2 =6
 3 * 3 =9
 3 * 4 =12
 3 * 5 =15
 3 * 6 =18
 3 * 7 =21
 3 * 8 =24
 3 * 9 =27
 3 * 10 =30
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

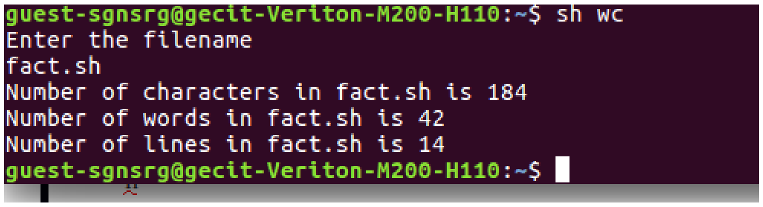
7. Write a shell script that counts the number of lines and words present in a given file.

Aim: To counts the number of lines and words present in a given file.

Description: wc command is used to display the number of characters, words, lines in a given file. “|” inputs the output of file content to wc.

Program:

```
echo Enter the filename
read file
w=`cat $file | wc -w`
c=`cat $file | wc -c`
l=`cat $file | wc -l`
echo Number of characters in $file is $c
echo Number of words in $file is $w
echo Number of lines in $file is $l
```



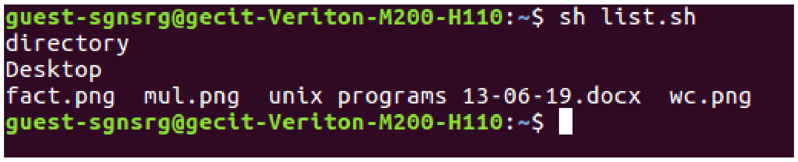
```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ sh wc
Enter the filename
fact.sh
Number of characters in fact.sh is 184
Number of words in fact.sh is 42
Number of lines in fact.sh is 14
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

8. Write a shell script that displays the list of all files in the given directory.

Aim: To display the list of all files in the given directory

Program:

```
echo directory
read d
for file in $d
do
if [ -d $file ]
then
ls $file
fi
done
```



```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ sh list.sh
directory
Desktop
fact.png  mul.png  unix programs 13-06-19.docx  wc.png
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

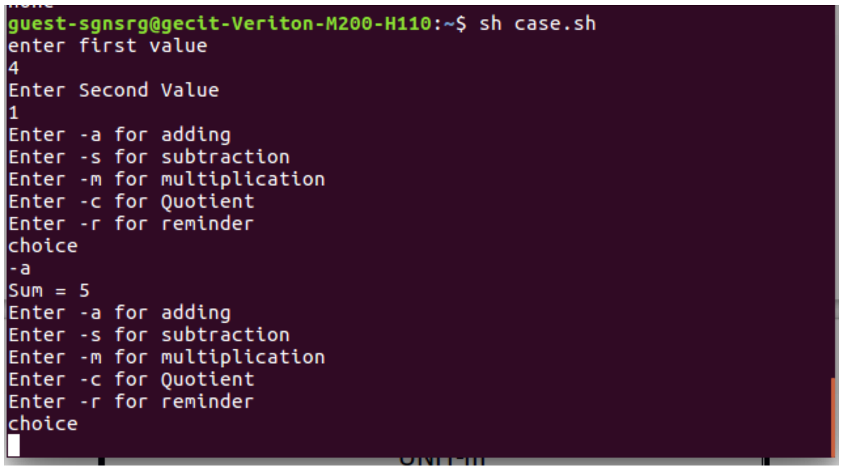
9. Write a shell script (small calculator) that adds, subtracts, multiplies and divides the given two integers. There are two division options: one returns the quotient and the other returns reminder. The script requires 3 arguments: The operation to be used and two integer numbers. The options are add (-a), subtract (-s), multiply (-m), quotient (-c) and reminder (-r).

Program:

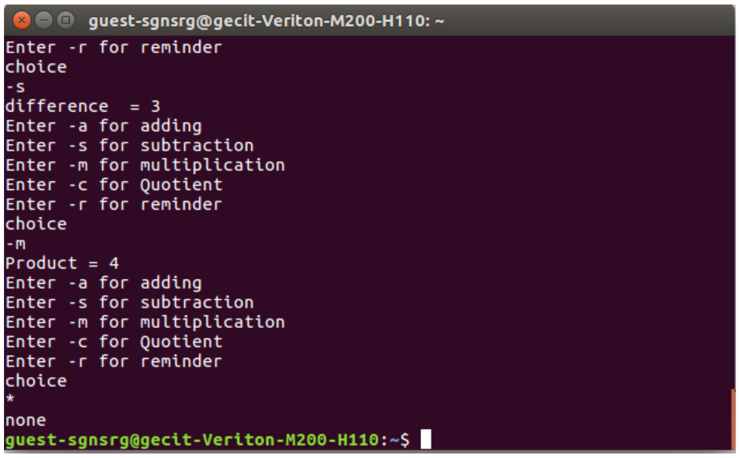
```
echo "enter first value"
read x
echo "Enter Second Value "
read y
while [ ${q:-1} -ne 0 ]
do
echo "Enter -a for adding"
echo "Enter -s for subtraction"
echo "Enter -m for multiplication"
echo "Enter -c for Quotient"
echo "Enter -r for reminder"
echo choice
read b
case $b in
-a)  p=`expr $x + $y`
echo "Sum = $p"
;;
-s)  p=`expr $x - $y`
echo "difference  = $p"

;;
-m)  p=`expr $x \* $y`
echo "Product = $p"
;;
-c)  p=`expr $x / $y`
echo "quotient = $p"
;;
-r)  p=`expr $x % $y`
echo "reminder = $p"
;;
*) echo "none"
break
```

```
;;
esac
done
```

Output:



```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ sh case.sh
enter first value
4
Enter Second Value
1
Enter -a for adding
Enter -s for subtraction
Enter -m for multiplication
Enter -c for Quotient
Enter -r for reminder
choice
-a
Sum = 5
Enter -a for adding
Enter -s for subtraction
Enter -m for multiplication
Enter -c for Quotient
Enter -r for reminder
choice
```



guest-sgnsrg@gecit-Veriton-M200-H110: ~

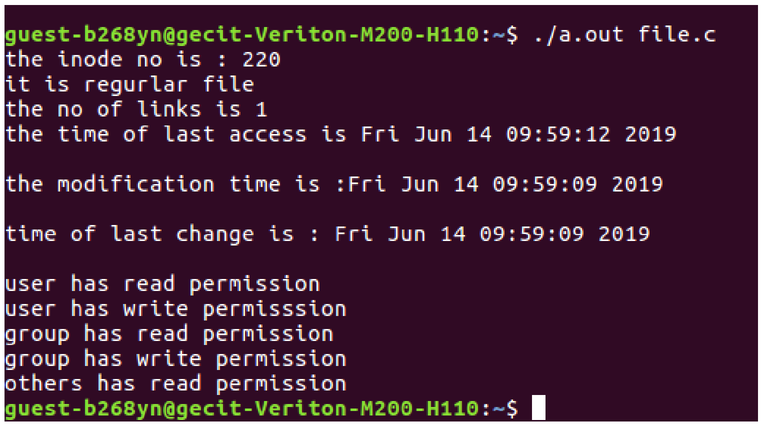
```
Enter -r for reminder
choice
-s
difference  = 3
Enter -a for adding
Enter -s for subtraction
Enter -m for multiplication
Enter -c for Quotient
Enter -r for reminder
choice
-m
Product = 4
Enter -a for adding
Enter -s for subtraction
Enter -m for multiplication
Enter -c for Quotient
Enter -r for reminder
choice
*
none
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

10. Write a program that takes one or more file/directory names as command line input and reports the following information on the file: File type, Number of links, Time of last access, Read, Write and Execute permissions.

Program:

```
#include<stdio.h>
#include<unistd.h>
#include<dirent.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<time.h>
int main(int argc,char *argv[])
{
char *at,*mt,*ct;
struct stat buf;
stat(argv[1],&buf);
printf("the inode no is : %ld\n",buf.st_ino);
if(S_ISDIR(buf.st_mode))
printf("it is a directory");
if(S_ISREG(buf.st_mode))
printf("it is regurlar file\n");
printf("the no of links is %ld\n",buf.st_nlink);
at=ctime(&buf.st_atime);
printf("the time of last access is %s\n",at);
mt=ctime(&buf.st_mtime);
printf("the modification time is :%s\n",mt);
ct=ctime(&buf.st_ctime);
printf("time of last change is : %s\n",ct);
if((buf.st_mode &S_IRUSR)==S_IRUSR)
printf("user has read permission\n");
if((buf.st_mode &S_IWUSR)==S_IWUSR)
printf("user has write permisssion\n");
if((buf.st_mode &S_IXUSR)==S_IXUSR)
printf("user has execute permission\n");
if((buf.st_mode &S_IRGRP)==S_IRGRP)
printf("group has read permission\n");
if((buf.st_mode &S_IWGRP)==S_IWGRP)
printf("group has write permission\n");
if((buf.st_mode &S_IXGRP)==S_IXGRP)
printf("group has execute permission\n");
if((buf.st_mode &S_IROTH)==S_IROTH)
printf("others has read permission\n");
if((buf.st_mode &S_IWOTH)==S_IWOTH)
printf("others has write permission\n");
if((buf.st_mode &S_IXOTH)==S_IXOTH)
printf("others has execute permisssion\n");
}
```

Output:



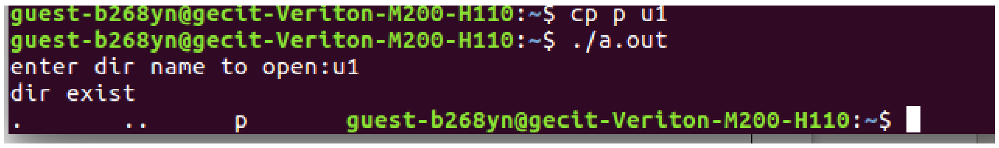
```
guest-b268yn@gecit-Veriton-M200-H110:~$ ./a.out file.c
the inode no is : 220
it is regurlar file
the no of links is 1
the time of last access is Fri Jun 14 09:59:12 2019

the modification time is :Fri Jun 14 09:59:09 2019

time of last change is : Fri Jun 14 09:59:09 2019

user has read permission
user has write permisssion
group has read permission
group has write permission
others has read permission
guest-b268yn@gecit-Veriton-M200-H110:~$
```

11. Write a C program that illustrates uses of the opendir, readdir, and closedir APIs.

Program:

```
#include<stdio.h>
#include<fcntl.h>
#include<dirent.h>
#include<unistd.h>
  int main()
  {
  char d[10];
  DIR *e;
  struct dirent *sd;
  printf("enter dir name to open:");
  scanf("%s",d);
  e=opendir(d);
  if(e==NULL)
  printf("dir does not exist");
  else
  {
  printf("dir exist\n");
  while((sd=readdir(e))!=NULL)
  {
  printf("%s\t", sd->d_name);
  }
  closedir(e);
  }
  }
```

Output:



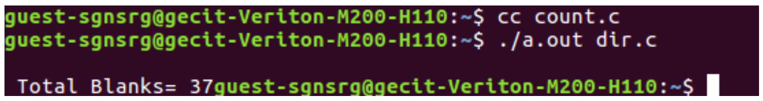
```
guest-b268yn@gecit-Veriton-M200-H110:~$ cp p u1
guest-b268yn@gecit-Veriton-M200-H110:~$ ./a.out
enter dir name to open:u1
dir exist
.       ..      p       guest-b268yn@gecit-Veriton-M200-H110:~$
```

12. Write a C program that counts the number of blanks in a text file using standard I/O

Program:

```
#include <fcntl.h>
#include <sys/stat.h>
#include <stdio.h>
int main(int argc, char **argv)
{
FILE *fd1;
int n,count=0;
char buf;
fd1=fopen(argv[1],"r");
while(!feof(fd1))
{
buf=fgetc(fd1);
if(buf=='')
count=count+1;
}
printf("\n Total Blanks= %d",count);
return (0);
}
```

Output:



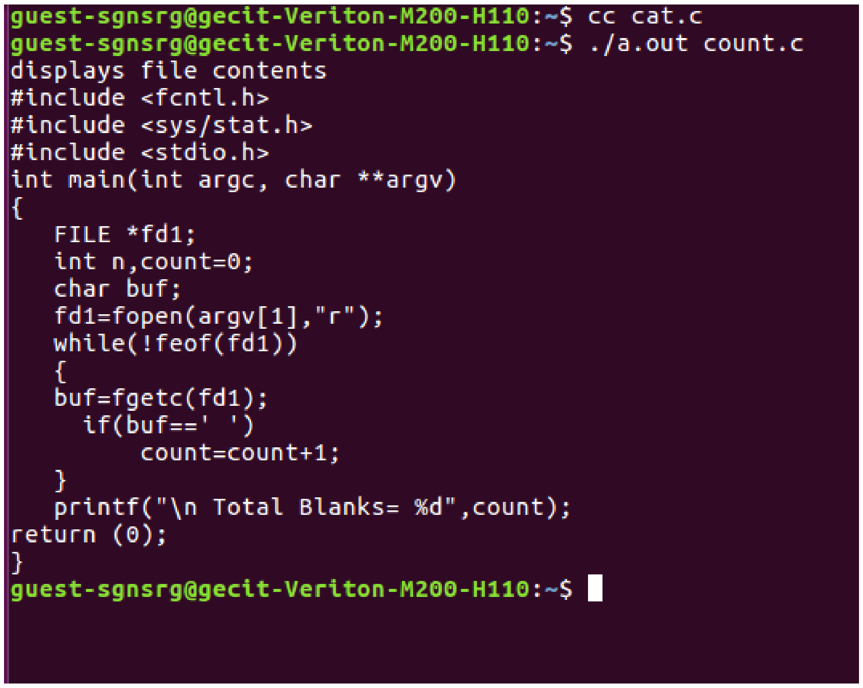
```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ cc count.c
guest-sgnsrg@gecit-Veriton-M200-H110:~$ ./a.out dir.c

 Total Blanks= 37guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

13. Implement in C, Unix command ‘cat’ using system calls.

Program:

```
#include<fcntl.h>
#include<sys/stat.h>
#include<stdio.h>
#include<unistd.h>
#define BUFSIZE 1
int main(int argc, char **argv)
{
int fd1;
int n;
char buf;
fd1=open(argv[1],O_RDONLY);
printf("displays file contents\n");
while((n=read(fd1,&buf,1))>0)
{
write(1,&buf,1);
}
return (0);
}
```

Output:



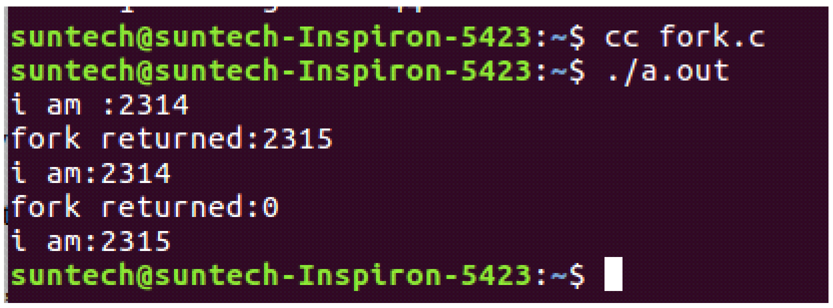
```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ cc cat.c
guest-sgnsrg@gecit-Veriton-M200-H110:~$ ./a.out count.c
displays file contents
#include <fcntl.h>
#include <sys/stat.h>
#include <stdio.h>
int main(int argc, char **argv)
{
   FILE *fd1;
   int n,count=0;
   char buf;
   fd1=fopen(argv[1],"r");
   while(!feof(fd1))
   {
   buf=fgetc(fd1);
     if(buf==' ')
         count=count+1;
   }
   printf("\n Total Blanks= %d",count);
return (0);
}
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

14. Write a C program that illustrates the creation of child process using fork system call.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
int main(int argc,char *argv[])
{
printf("i am :%d¥n",(int) getpid());
pid_t pid=fork();
printf("fork returned:%d¥n", (int) pid);
printf("i am:%d¥n",(int)getpid());
return 0;
}
```

output:



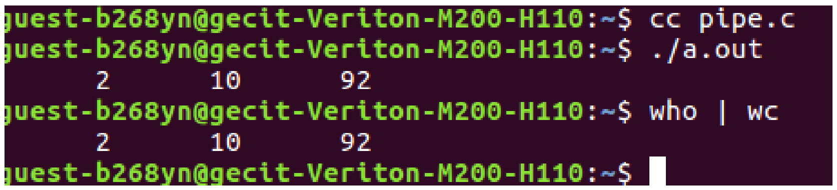
```
suntech@suntech-Inspiron-5423:~$ cc fork.c
suntech@suntech-Inspiron-5423:~$ ./a.out
i am :2314
fork returned:2315
i am:2314
fork returned:0
i am:2315
suntech@suntech-Inspiron-5423:~$
```

15. Write a C program that illustrates how to execute two commands concurrently with a command pipe.

Program:

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/types.h>
int main()
{
int pfd[2],p;
pipe(pfd);
p=fork();
if(p==0)//for child
{
close(pfd[0]);
close(1);
dup(pfd[1]);
execlp("who","who", (char *)0);
}
else
{
close(pfd[1]);
close(0);
dup(pfd[0]);
execlp("wc","wc",(char *)0);
}
}
```

output:



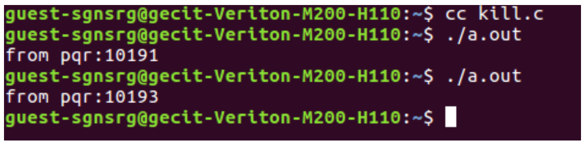
```
uest-b268yn@gecit-Veriton-M200-H110:~$ cc pipe.c
uest-b268yn@gecit-Veriton-M200-H110:~$ ./a.out
      2      10      92
uest-b268yn@gecit-Veriton-M200-H110:~$ who | wc
      2      10      92
uest-b268yn@gecit-Veriton-M200-H110:~$
```

16. Write a C program to implement the Kill function.

Program:

```
#include<signal.h>
#include<stdio.h>
#include<unistd.h>
void xyz(int n)
{
printf("hello: %d\n", getpid());
signal(2,xyz);
}
void pqr(int n)
{
printf("from pqr:%d\n",getpid());
}
int main()
{
signal(2,xyz);
if(fork()==0)
{
signal(2,pqr);
kill(0,2);
}
while(1)
return 0;
}
```

Output:



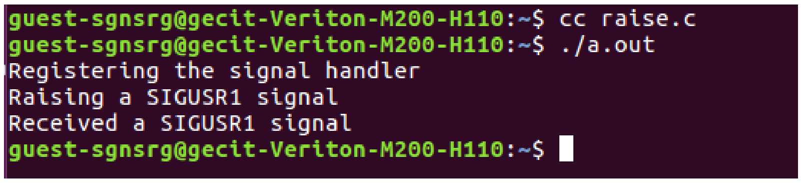
```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ cc kill.c
guest-sgnsrg@gecit-Veriton-M200-H110:~$ ./a.out
from pqr:10191
guest-sgnsrg@gecit-Veriton-M200-H110:~$ ./a.out
from pqr:10193
guest-sgnsrg@gecit-Veriton-M200-H110:~$ 
```

17.Write a C program to implement the raise function.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
void signal_handler(int signal)
{
/* Display a message indicating we have received a signal */
if (signal == SIGUSR1)
printf("Received a SIGUSR1 signal\n");
/* Exit the application */
exit(0);
}
int main(int argc, const char * argv[])
{
/* Display a message indicating we are registering the signal handler */
printf("Registering the signal handler\n");
/* Register the signal handler */
signal(SIGUSR1, signal_handler);
/* Display a message indicating we are raising a signal */
printf("Raising a SIGUSR1 signal\n");
/* Raise the SIGUSR1 signal */
raise(SIGUSR1);
/* Display a message indicating we are leaving main */
printf("Finished main\n");
return 0;
}
```

Output:



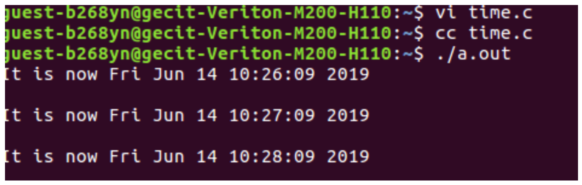
```
guest-sgnsrg@gecit-Veriton-M200-H110:~$ cc raise.c
guest-sgnsrg@gecit-Veriton-M200-H110:~$ ./a.out
Registering the signal handler
Raising a SIGUSR1 signal
Received a SIGUSR1 signal
guest-sgnsrg@gecit-Veriton-M200-H110:~$
```

18. Write a C program that displays the real time of a day every 60 seconds.

Program:

```
#include<stdio.h>
 #include<sys/time.h>
 #include<sys/signal.h>
#include<unistd.h>
/* Declarations */
void main();
 int times_up();

 void main()
        {
          for (; ;)
    {
      times_up(1);
    sleep(60);
    }
        }
    int times_up(sig)
        int sig;
        {
          long now;
          struct tms *ptr;
          long time();
            char *ctime();
           time(&now);
          printf("It is now %s\n", ctime (&now));
           return (sig);
        }
```

Output:



```
guest-b268yn@gecit-Veriton-M200-H110:~$ vi time.c
guest-b268yn@gecit-Veriton-M200-H110:~$ cc time.c
guest-b268yn@gecit-Veriton-M200-H110:~$ ./a.out
It is now Fri Jun 14 10:26:09 2019

It is now Fri Jun 14 10:27:09 2019

It is now Fri Jun 14 10:28:09 2019
```

Additional Programs:

1. Write a shell script to display files which has read, write and execute permissions.

   Program:

```
echo “enter a filename”
read  f
for f in `ls`
do
          if [ -w $f –a  -r $f –a –x $f ]
          then
          echo $f
          fi
done
```

2. Write a shell script to display the biggest number in given input numbers.

   Program:

```
echo “enter a num”
read num
if  [ $1 –ge $2  ]
then
          if  [ $1 –ge $3  ]
          then
     echo $1 “is biggest num”
          else
     echo $3 “is biggest num”
fi
elif  [  $2 –ge $3 ]
then
echo $2 “is bigger num”
else
echo “Biggest num is “ $3
          fi
```

3. Write a C program that illustrates the creation of child process using fork system call. One process finds sum of even series and other process finds sum of odd series.

   Program:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
int main()
{
int i,n,sum=0;
pid_t pid;
system(“clear”);
printf(“Enter n value:”);
scanf(“%d”,&n)
pid=fork();
if(pid==0)
{
printf(“From child process\n”);
for(i=1;i<n;i+=2)
{
printf(“%d\”,i);
sum+=i;
}
printf("Odd sum:%d\n",sum);
}
else
{
printf("From process\n");
for(i=0;i<n;i+=2)
{
printf("%d\",i);
sum+=i;
}
printf("Even sum:%d\n",sum);
}
}
```