EmbedUR Systems Privated Limited

Linux program Training

Module - 1

By:

A.Buvaneshkumaar

Mepco Schlenk Engineering College

Sivakasi,Virudhunagar District.,

Basic Commands

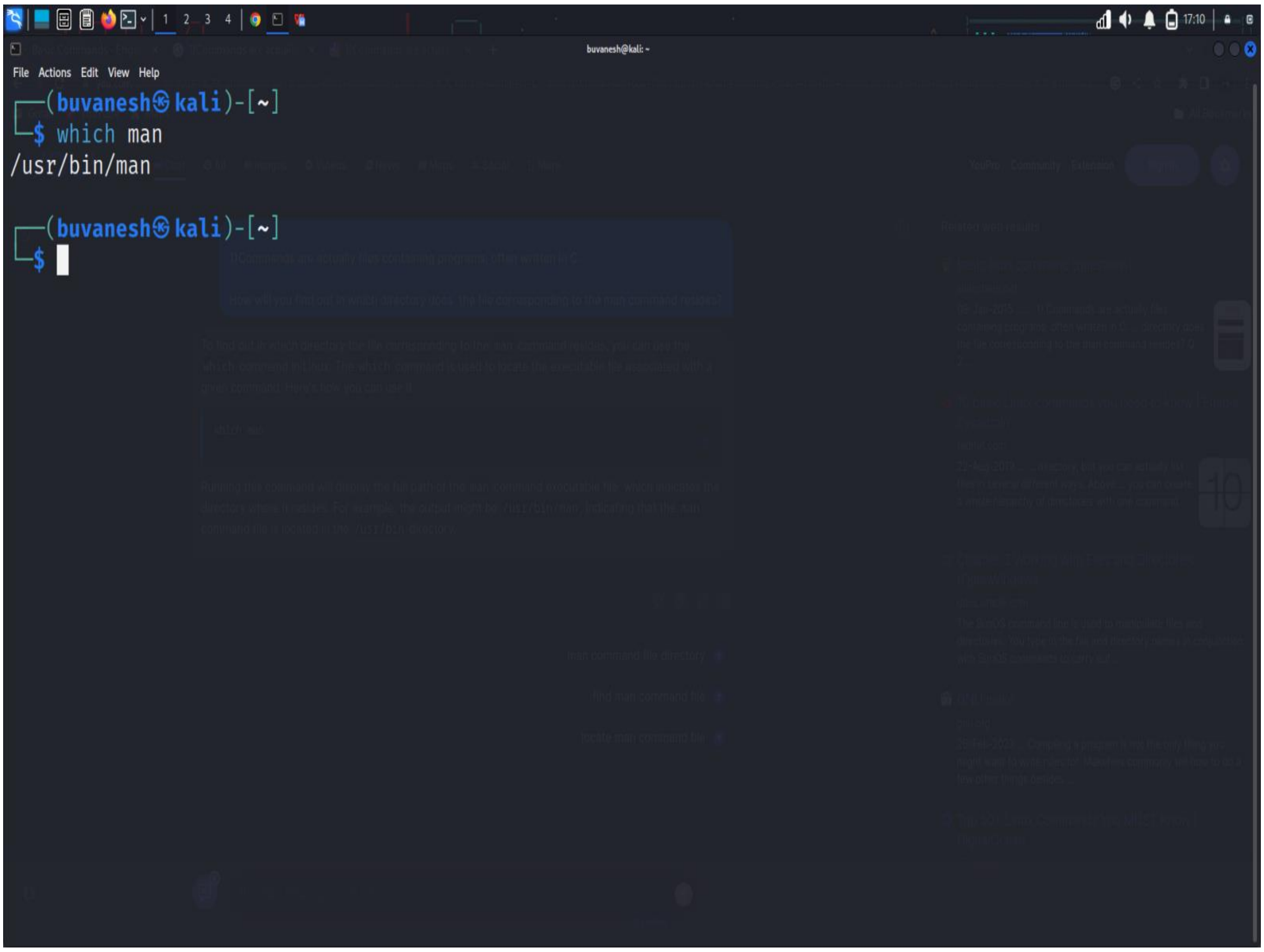
1) Commands are actually files containing programs, often written in C.

How will you find out in which directory does the file corresponding to the man command resides?

To find out in which directory the file corresponding to the man command resides, you can use the **which** command in Linux. The which command is used to locate the executable file associated with a given command

Command: which man

Output:



```
┌──(buvanesh㉿kali)-[~]
└─$ which man
/usr/bin/man

┌──(buvanesh㉿kali)-[~]
└─$ 
```

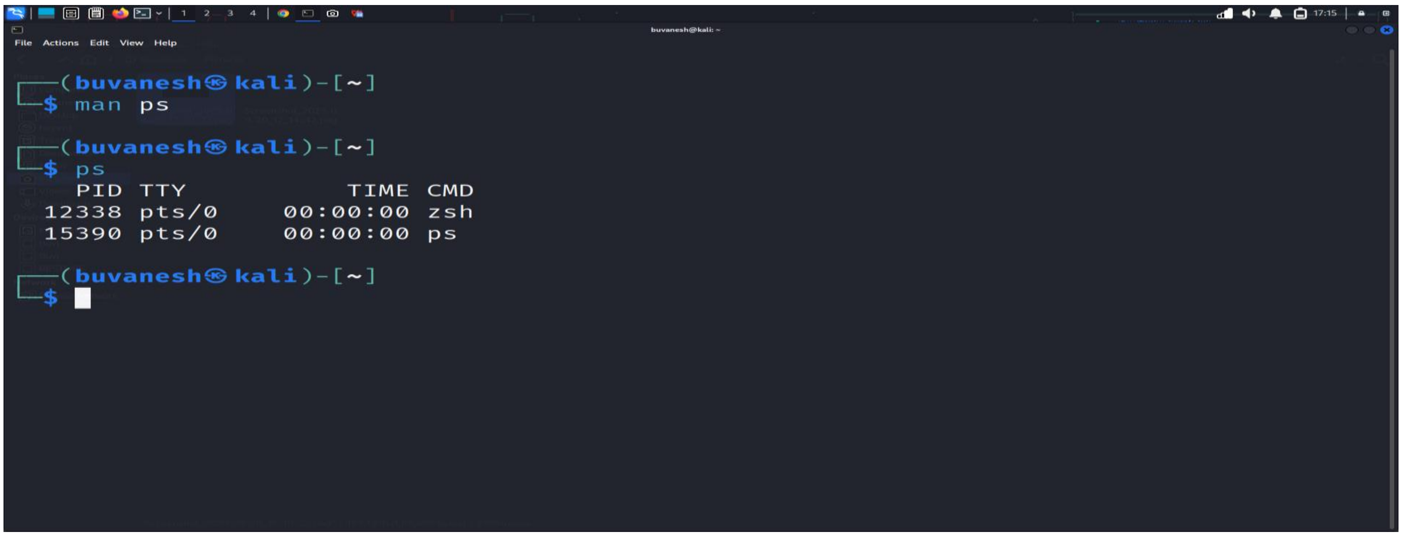
2) How will you find out what is the use of the ps command.

Running this command will display the manual page for the **ps** command, which includes a description of what the command does, the available options and their usage, examples, and other relevant information.

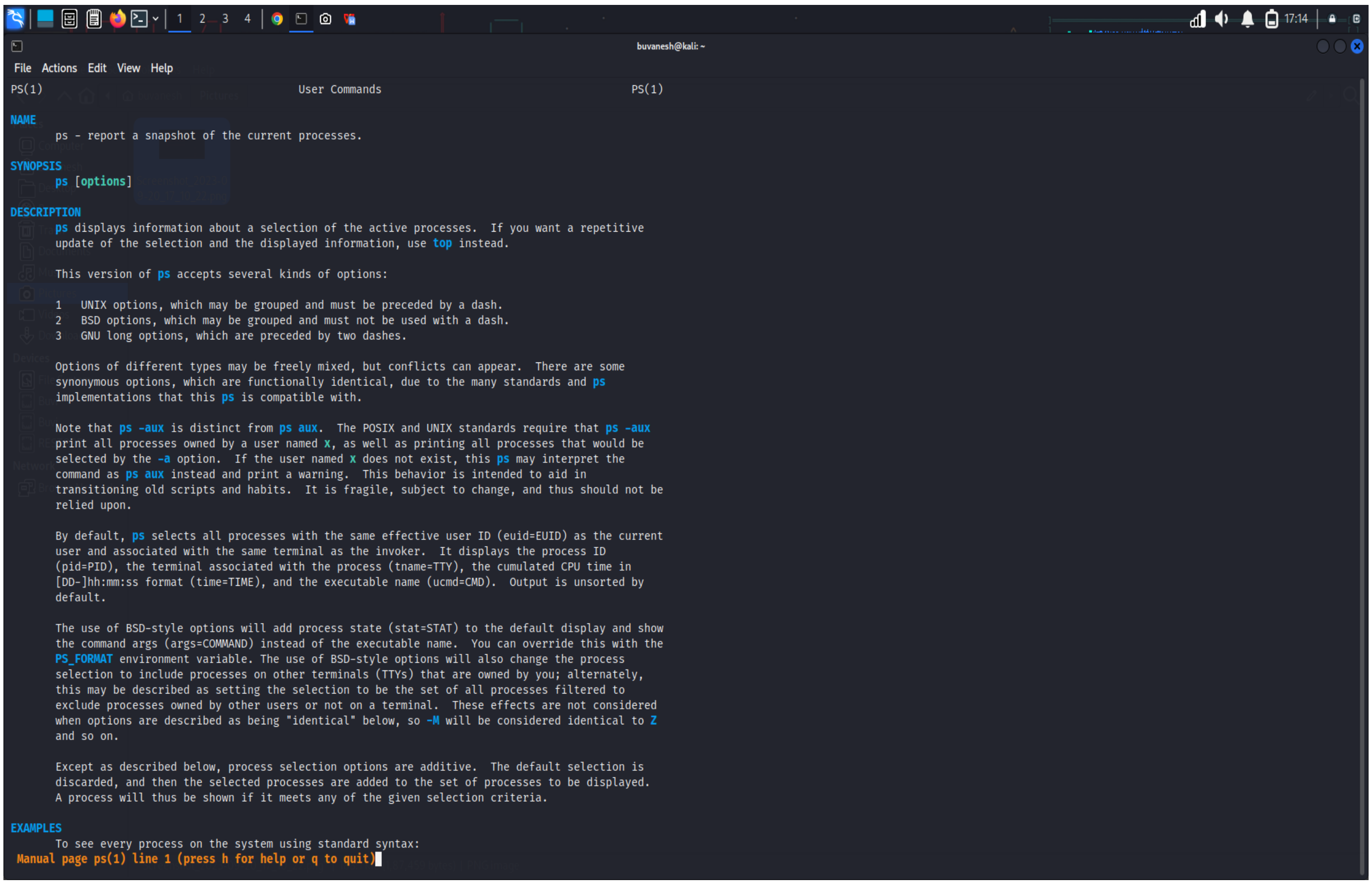
Command : ps

Command : ps man

Output:



```
┌──(buvanesh㉿kali)-[~]
└─$ man ps

┌──(buvanesh㉿kali)-[~]
└─$ ps
    PID TTY          TIME CMD
  12338 pts/0    00:00:00 zsh
  15390 pts/0    00:00:00 ps

┌──(buvanesh㉿kali)-[~]
└─$ 
```



```
PS(1)                            User Commands                           PS(1)

NAME
       ps - report a snapshot of the current processes.

SYNOPSIS
       ps [options]

DESCRIPTION
       ps displays information about a selection of the active processes.  If you want a repetitive
       update of the selection and the displayed information, use top instead.

       This version of ps accepts several kinds of options:

       1   UNIX options, which may be grouped and must be preceded by a dash.
       2   BSD options, which may be grouped and must not be used with a dash.
       3   GNU long options, which are preceded by two dashes.

       Options of different types may be freely mixed, but conflicts can appear.  There are some
       synonymous options, which are functionally identical, due to the many standards and ps
       implementations that this ps is compatible with.

       Note that ps -aux is distinct from ps aux.  The POSIX and UNIX standards require that ps -aux
       print all processes owned by a user named x, as well as printing all processes that would be
       selected by the -a option.  If the user named x does not exist, this ps may interpret the
       command as ps aux instead and print a warning.  This behavior is intended to aid in
       transitioning old scripts and habits.  It is fragile, subject to change, and thus should not be
       relied upon.

       By default, ps selects all processes with the same effective user ID (euid=EUID) as the current
       user and associated with the same terminal as the invoker.  It displays the process ID
       (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in
       [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD).  Output is unsorted by
       default.

       The use of BSD-style options will add process state (stat=STAT) to the default display and show
       the command args (args=COMMAND) instead of the executable name.  You can override this with the
       PS_FORMAT environment variable. The use of BSD-style options will also change the process
       selection to include processes on other terminals (TTYs) that are owned by you; alternately,
       this may be described as setting the selection to be the set of all processes filtered to
       exclude processes owned by other users or not on a terminal.  These effects are not considered
       when options are described as being "identical" below, so -M will be considered identical to Z
       and so on.

       Except as described below, process selection options are additive.  The default selection is
       discarded, and then the selected processes are added to the set of processes to be displayed.
       A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES
       To see every process on the system using standard syntax:
 Manual page ps(1) line 1 (press h for help or q to quit)
```

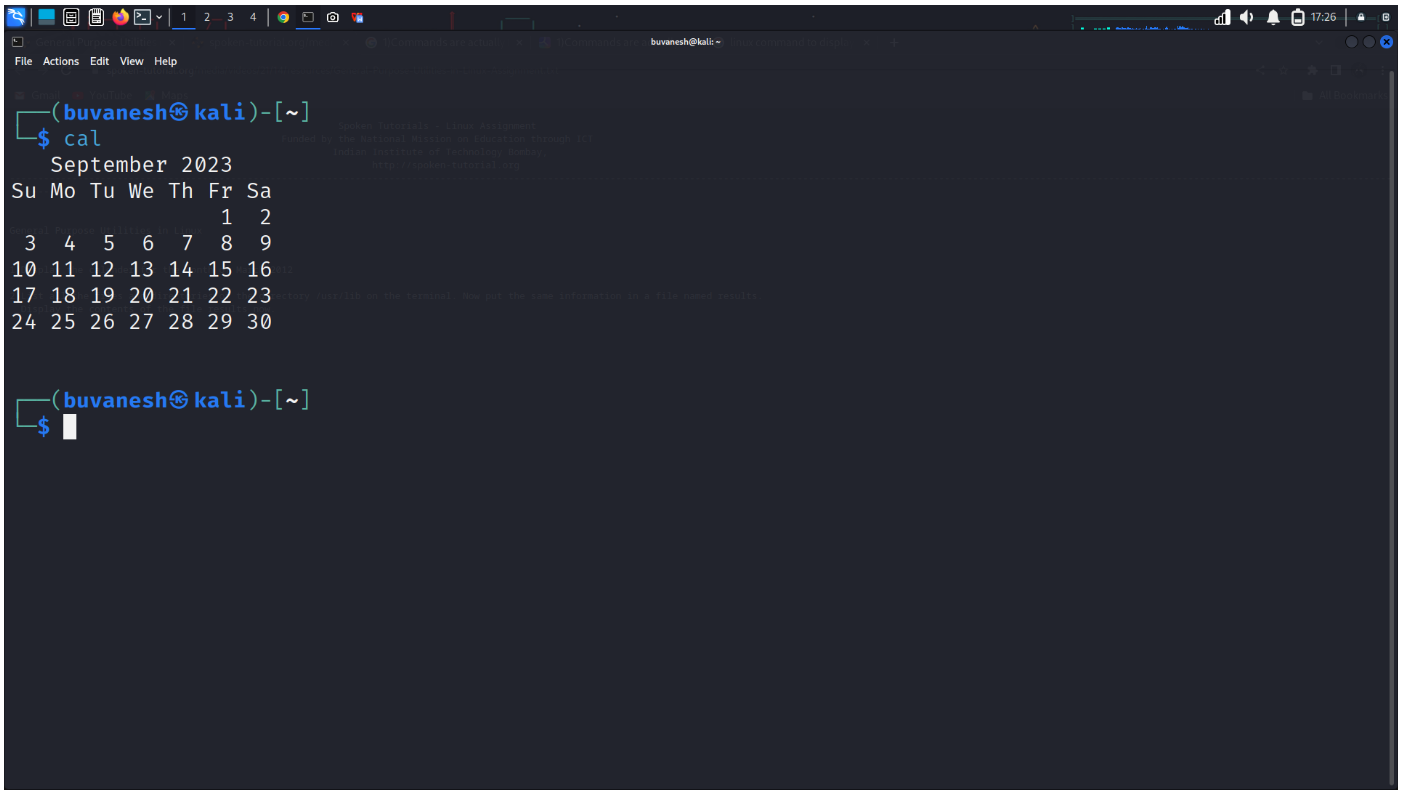
General purpose utilities in Linux

1) Display the calender for the month of March 2012

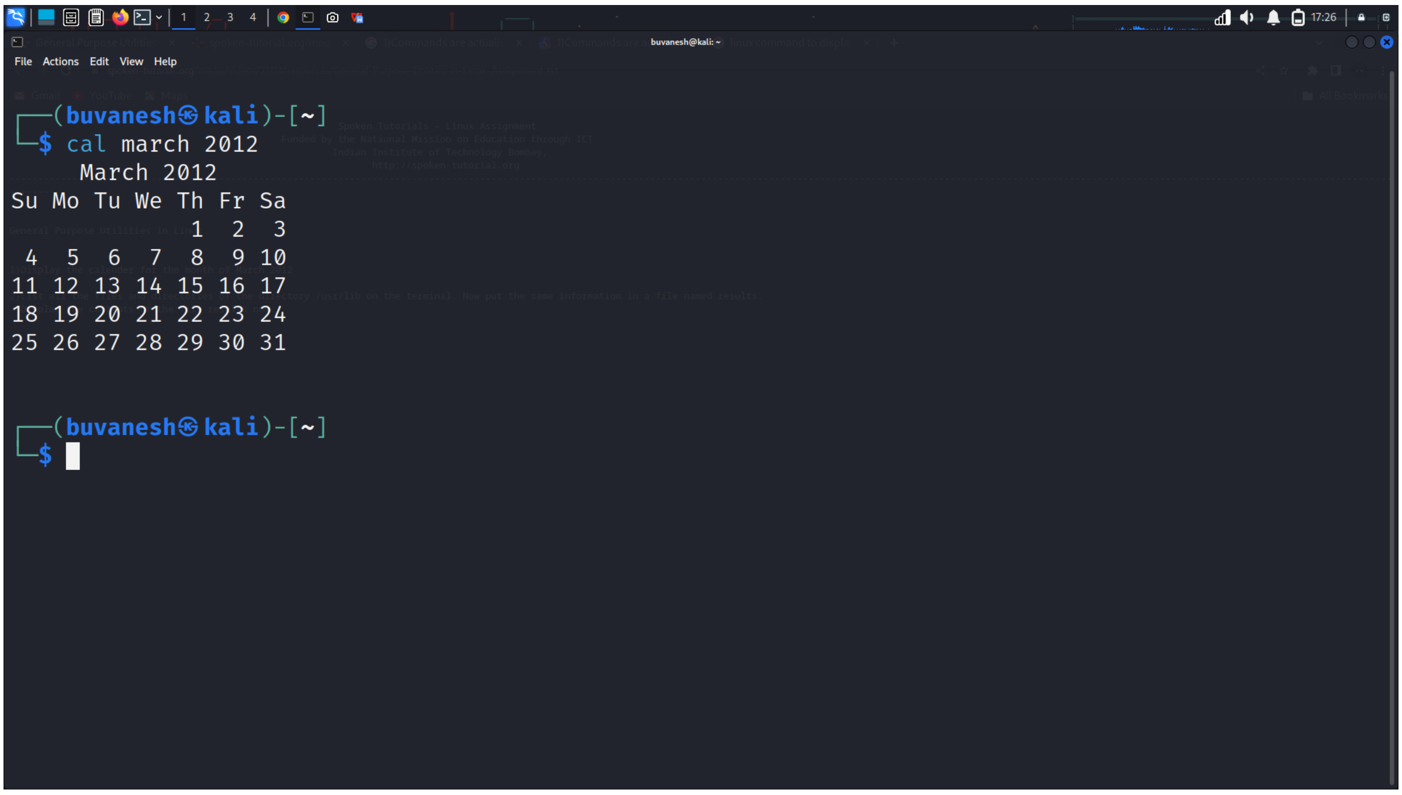
Command : cal (used to view the current calender)

Command : cal march 2012

Output:



```
┌──(buvanesh㉿kali)-[~]
└─$ cal
   September 2023
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

┌──(buvanesh㉿kali)-[~]
└─$
```



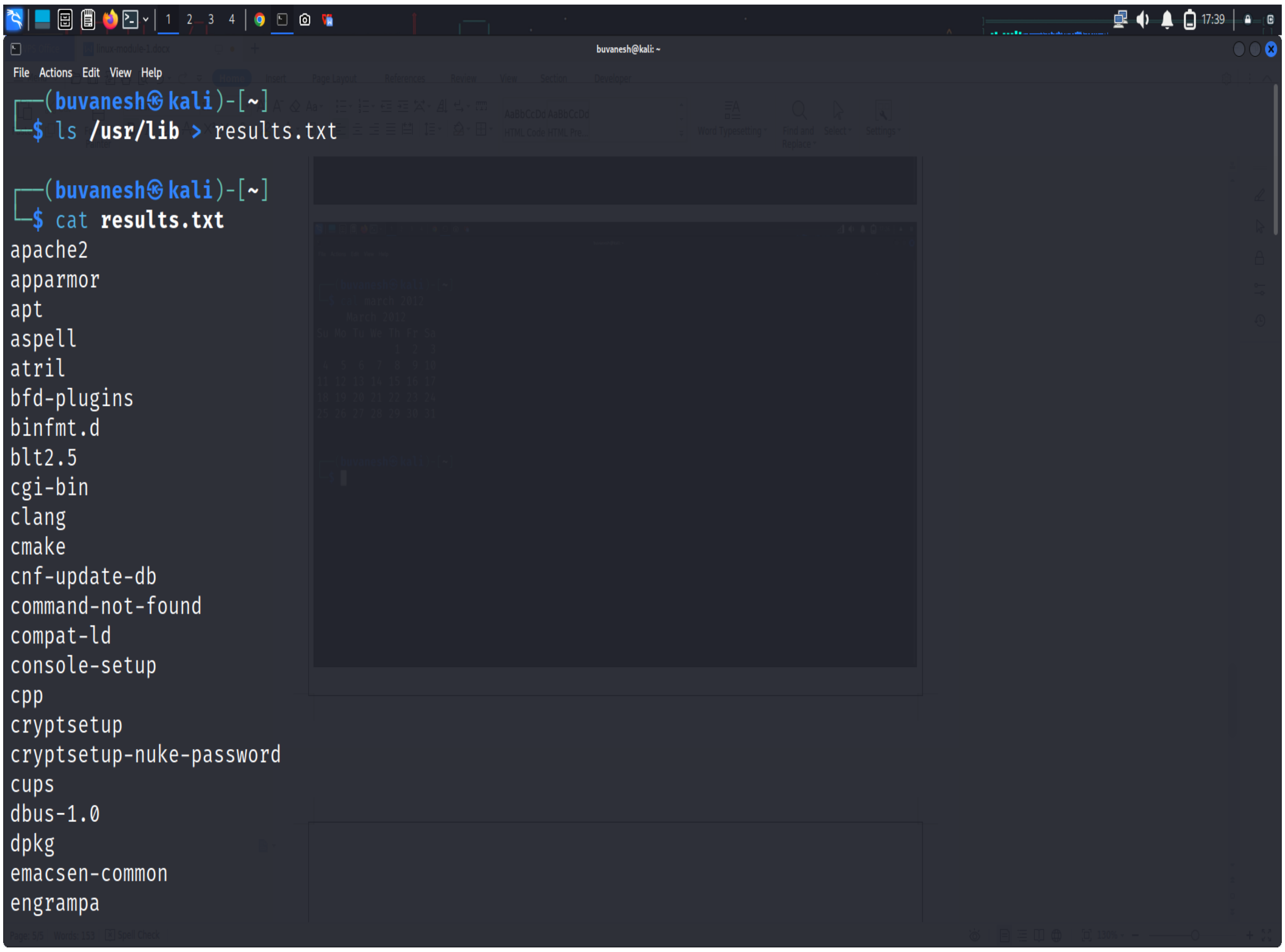
```
┌──(buvanesh㉿kali)-[~]
└─$ cal march 2012
     March 2012
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

┌──(buvanesh㉿kali)-[~]
└─$
```

2)List all the files and directories of the directory /usr/lib on the terminal. Now put the same information in a file named results. Display the contents of the file results now.

Command : ls /usr/lib > results.txt

Command : cat results.txt

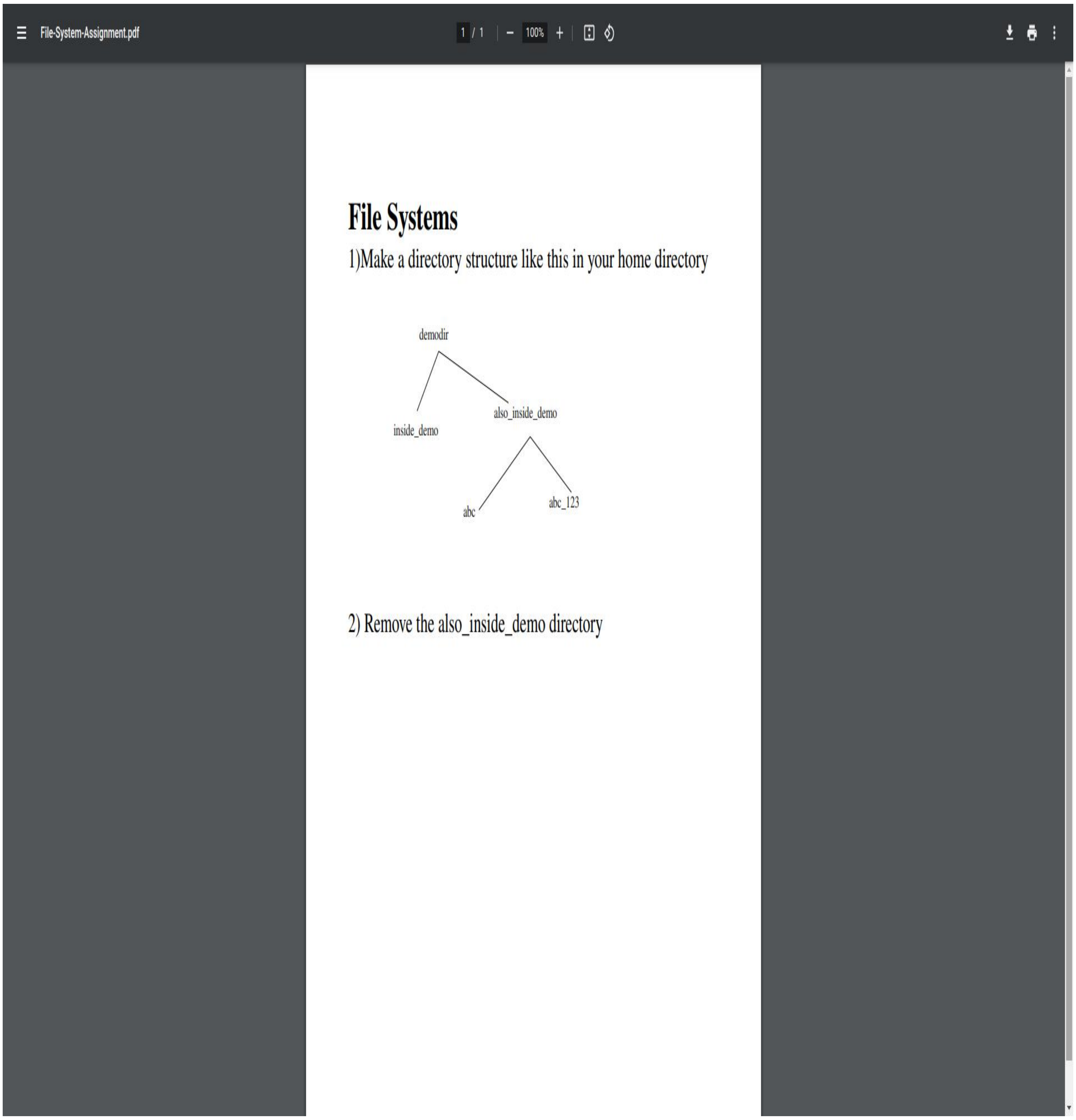
Output:



```
┌──(buvanesh㉿kali)-[~]
└─$ ls /usr/lib > results.txt

┌──(buvanesh㉿kali)-[~]
└─$ cat results.txt
apache2
apparmor
apt
aspell
atril
bfd-plugins
binfmt.d
blt2.5
cgi-bin
clang
cmake
cnf-update-db
command-not-found
compat-ld
console-setup
cpp
cryptsetup
cryptsetup-nuke-password
cups
dbus-1.0
dpkg
emacsen-common
engrampa
```

File System

1)Question



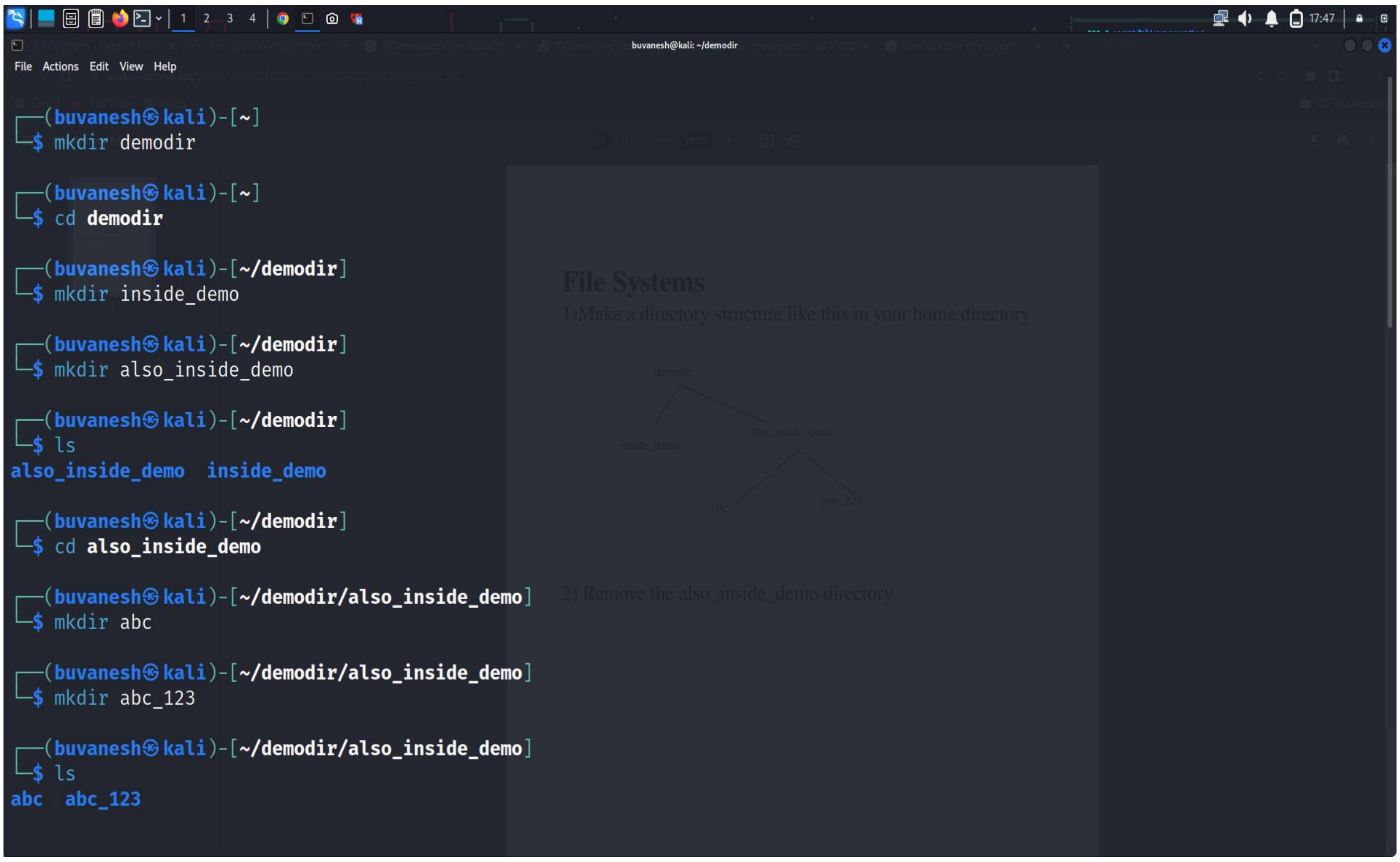
File-System-Assignment.pdf

File Systems

1)Make a directory structure like this in your home directory

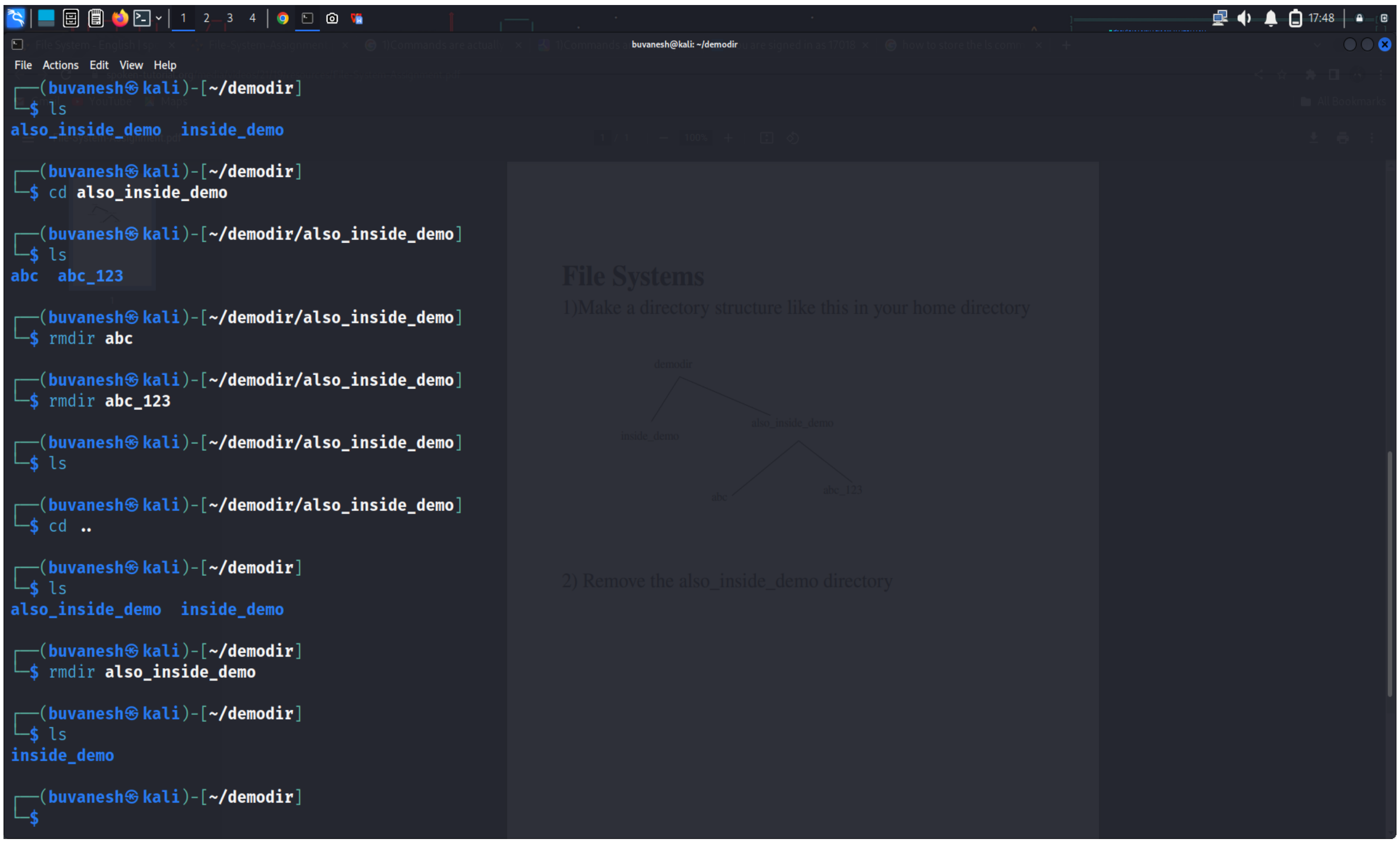
2) Remove the also_inside_demo directory

Output:



```
┌──(buvanesh㉿kali)-[~]
└─$ mkdir demodir

┌──(buvanesh㉿kali)-[~]
└─$ cd demodir

┌──(buvanesh㉿kali)-[~/demodir]
└─$ mkdir inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ mkdir also_inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ ls
also_inside_demo  inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ cd also_inside_demo

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ mkdir abc

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ mkdir abc_123

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ ls
abc  abc_123
```

2)Output:



```
┌──(buvanesh㉿kali)-[~/demodir]
└─$ ls
also_inside_demo  inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ cd also_inside_demo

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ ls
abc  abc_123

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ rmdir abc

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ rmdir abc_123

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ ls

┌──(buvanesh㉿kali)-[~/demodir/also_inside_demo]
└─$ cd ..

┌──(buvanesh㉿kali)-[~/demodir]
└─$ ls
also_inside_demo  inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ rmdir also_inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$ ls
inside_demo

┌──(buvanesh㉿kali)-[~/demodir]
└─$
```

File Attributes

1) Create a file abc.txt and change the ownership of this file to some other user on your machine,and also change the group to family.

Command : sudo useradd kumar

Command : sudo passwd kumar123

Command : sudo usermod -aG sudo kumar

Command : sudo chown kumar abc.txt

Command : sudo chgrp family abc.txt

2)Create a file exercise.txt and make it executable.

Command : vi exercise.txt (creation of file)

Command : chmod +x exercise.txt (change the permission)

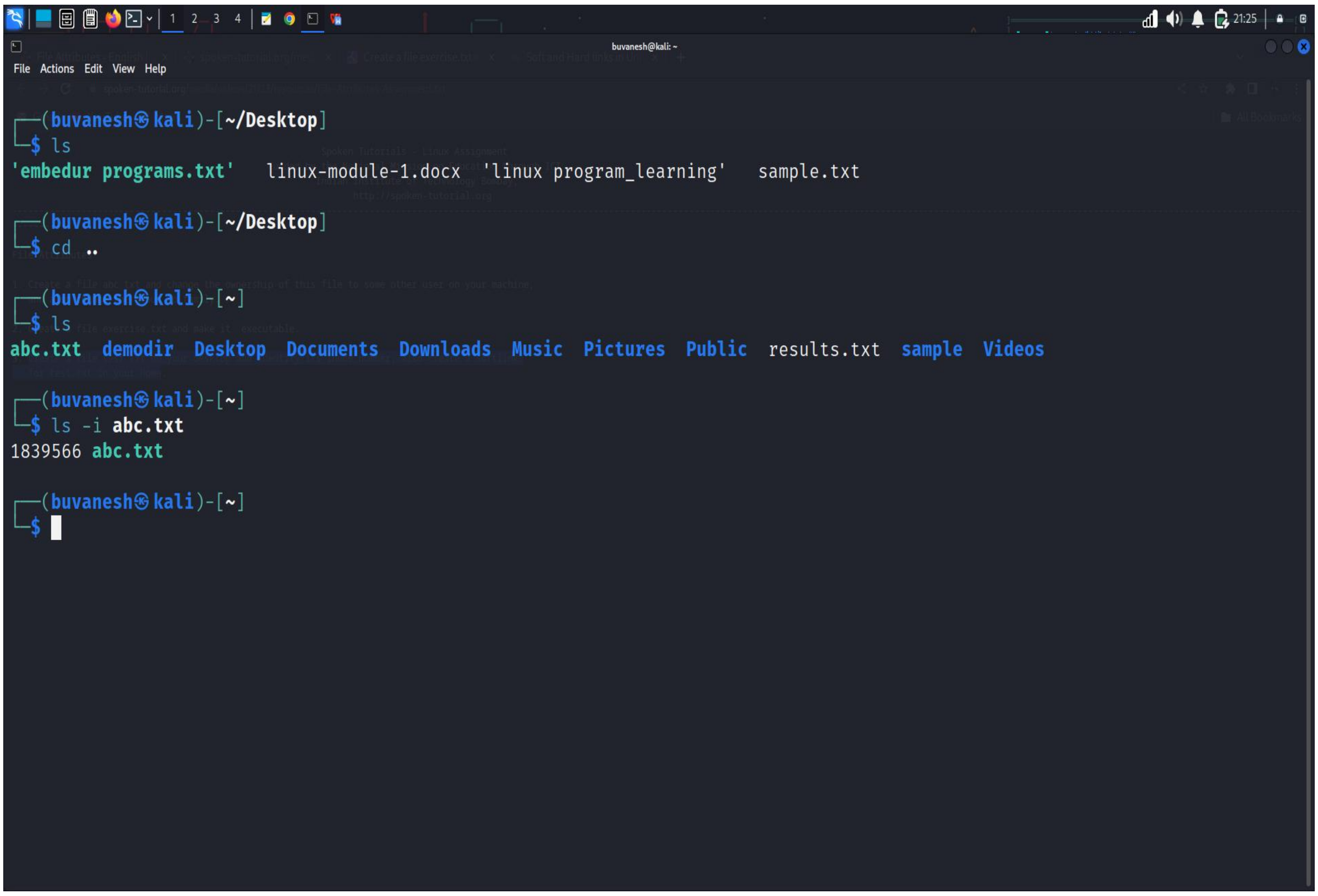
3)Create a file test.txt on your desktop and identity its inode number ,also create a softlink for text.txt in your home

Command : vi text.txt (creation of file)

Command : ls -i text.txt (To identify the inode number)

Command : ln -s /home/buvanesh/text.txt text.txt (creation of softlink)

Output:



```
┌──(buvanesh㉿kali)-[~/Desktop]
└─$ ls
'embedur programs.txt'   linux-module-1.docx  'linux program_learning'   sample.txt

┌──(buvanesh㉿kali)-[~/Desktop]
└─$ cd ..

┌──(buvanesh㉿kali)-[~]
└─$ ls
abc.txt  demodir  Desktop  Documents  Downloads  Music  Pictures  Public  results.txt  sample  Videos

┌──(buvanesh㉿kali)-[~]
└─$ ls -i abc.txt
1839566 abc.txt

┌──(buvanesh㉿kali)-[~]
└─$ 
```

Redirection pipes

1) Create a file name error_log in your current directory. Suppose you do not have any file named aa11 in your current directory.

How can you redirect the error message to the file error_log when we apply the command "wc -l aa11" ?

How can you ensure that all the error log are appended to the error_log file?

Command : touch error_log && wc -l aa11 2>> error_log

Description :

- touch error_log - creates a file in the directory
- wc -l aa11 - count the number of lines in the file aa11
- 2>> error_log - redirectes the error message

2)Create files named test1, test2, testa, testb

How can you count the number of files starting with test and then having only one digit in their name using only a single line command ?

Command : vi test (creation of file)

Command : ls -ld test[0-9] | wc -l

Description :

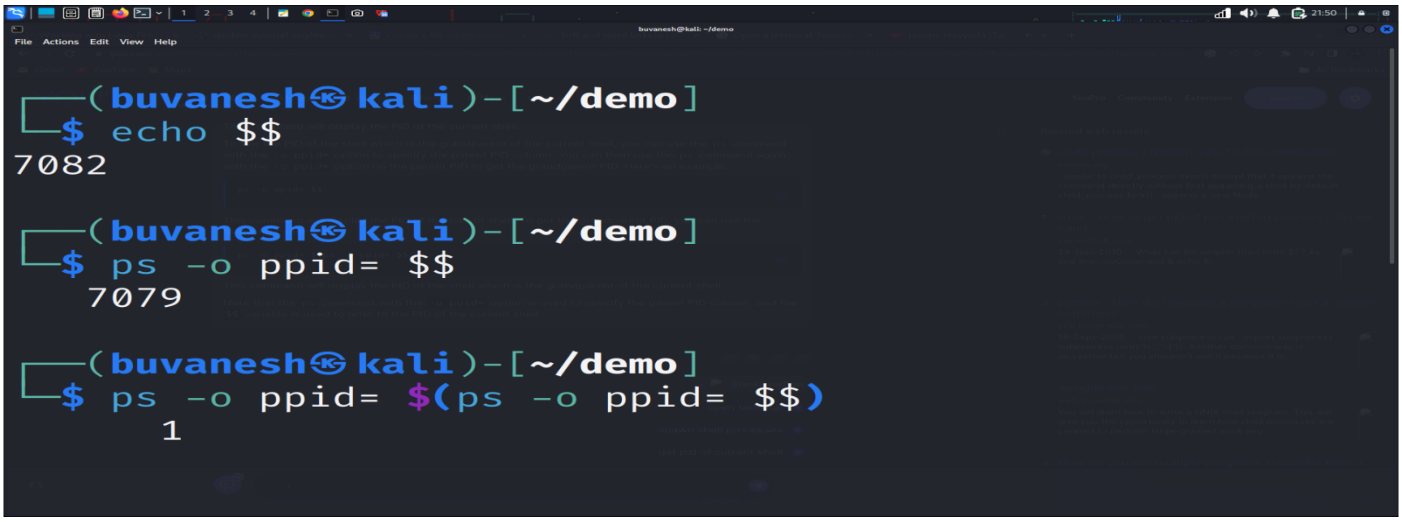
- |- pipe symbol, which redirects the output of the previous command to the input of the next command.
- wc -l: This command counts the number of lines in its input. By piping the output of ls to wc -l, we can count the number of files starting with "test" and having only one digit in their name.

Linux Process

1) Open a terminal. Now spawn three shell processes one after another i.e. first spawn one shell, then from the spawned shell, spawn one new shell and so on. Now,

how can you see the PID of the current shell ? How can you see the PID of the shell which is the grandparent of the current shell?

Output:



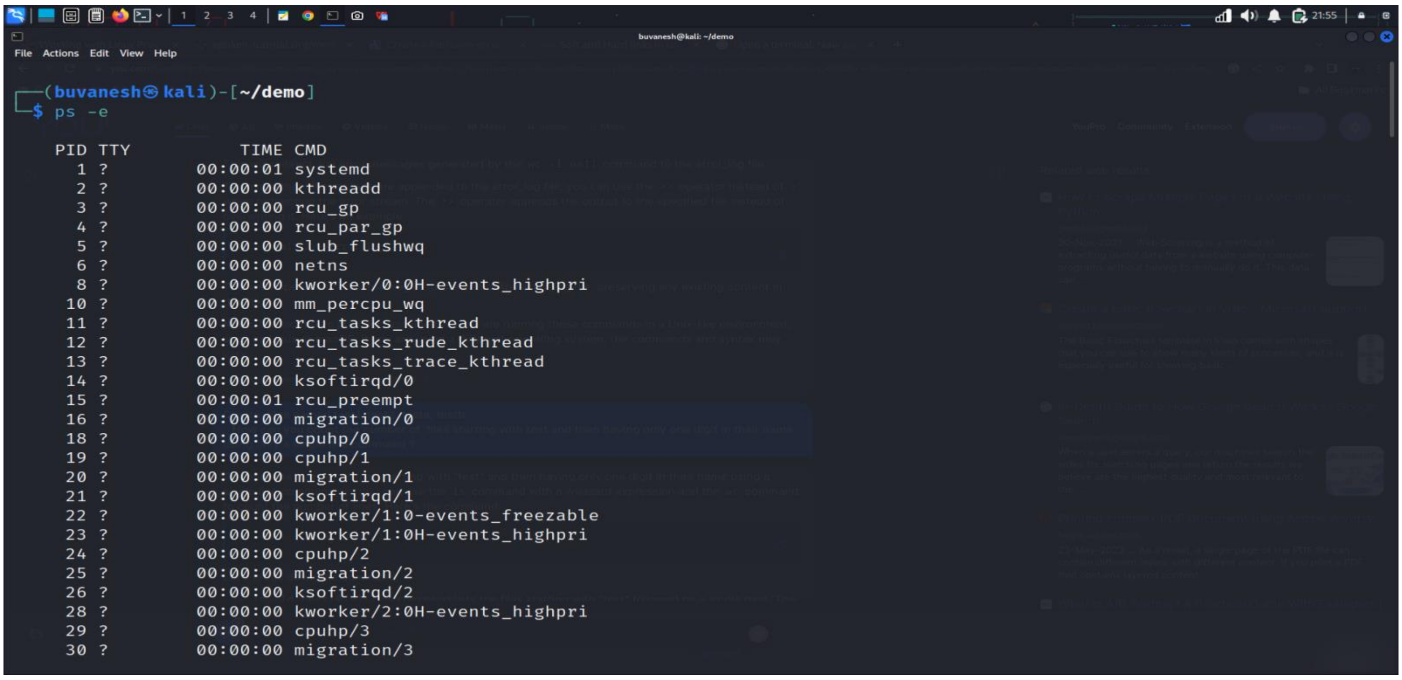
```
┌──(buvanesh㉿kali)-[~/demo]
└─$ echo $$
7082

┌──(buvanesh㉿kali)-[~/demo]
└─$ ps -o ppid= $$
   7079

┌──(buvanesh㉿kali)-[~/demo]
└─$ ps -o ppid= $(ps -o ppid= $$)
      1
```

2) How can you see all the processes (both system & user processes) in your computer?

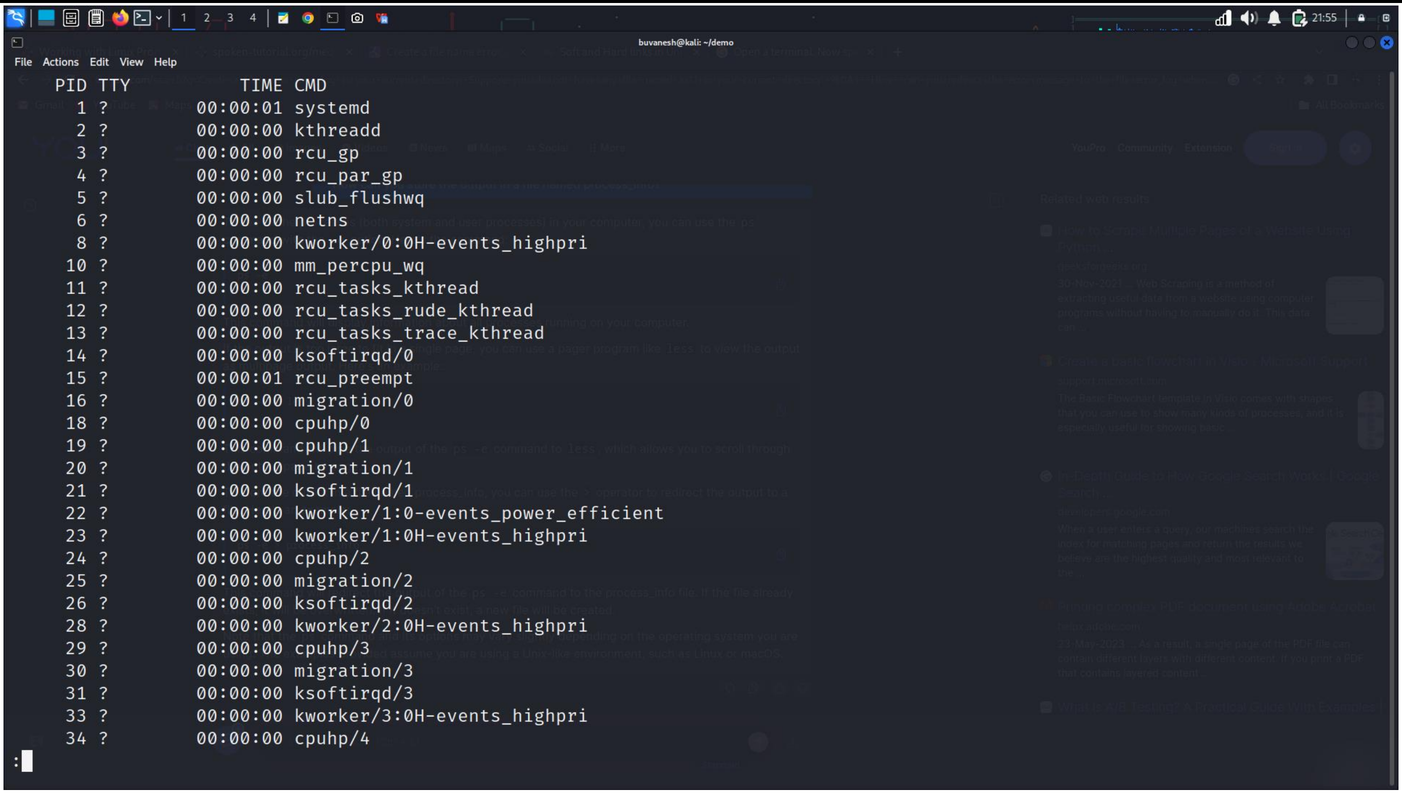
The output can be quite large. How can you view the output as multipage output ?

How can you store the output in a file named process_info?

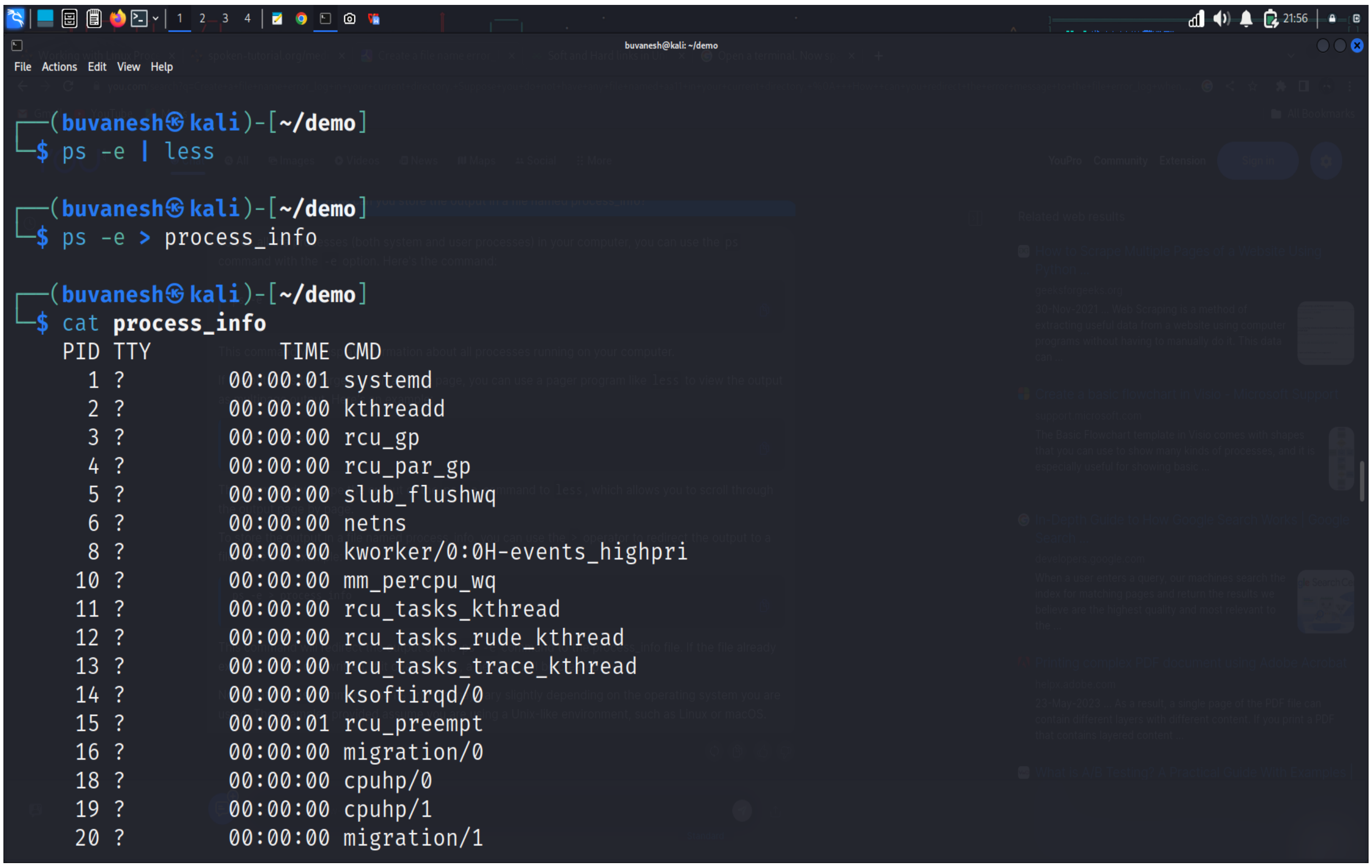
Output:



```
┌──(buvanesh㉿kali)-[~/demo]
└─$ ps -e

    PID TTY          TIME CMD
      1 ?        00:00:01 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_kthread
     12 ?        00:00:00 rcu_tasks_rude_kthread
     13 ?        00:00:00 rcu_tasks_trace_kthread
     14 ?        00:00:00 ksoftirqd/0
     15 ?        00:00:01 rcu_preempt
     16 ?        00:00:00 migration/0
     18 ?        00:00:00 cpuhp/0
     19 ?        00:00:00 cpuhp/1
     20 ?        00:00:00 migration/1
     21 ?        00:00:00 ksoftirqd/1
     22 ?        00:00:00 kworker/1:0-events_freezable
     23 ?        00:00:00 kworker/1:0H-events_highpri
     24 ?        00:00:00 cpuhp/2
     25 ?        00:00:00 migration/2
     26 ?        00:00:00 ksoftirqd/2
     28 ?        00:00:00 kworker/2:0H-events_highpri
     29 ?        00:00:00 cpuhp/3
     30 ?        00:00:00 migration/3
```



```
    PID TTY          TIME CMD
      1 ?        00:00:01 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_kthread
     12 ?        00:00:00 rcu_tasks_rude_kthread
     13 ?        00:00:00 rcu_tasks_trace_kthread
     14 ?        00:00:00 ksoftirqd/0
     15 ?        00:00:01 rcu_preempt
     16 ?        00:00:00 migration/0
     18 ?        00:00:00 cpuhp/0
     19 ?        00:00:00 cpuhp/1
     20 ?        00:00:00 migration/1
     21 ?        00:00:00 ksoftirqd/1
     22 ?        00:00:00 kworker/1:0-events_power_efficient
     23 ?        00:00:00 kworker/1:0H-events_highpri
     24 ?        00:00:00 cpuhp/2
     25 ?        00:00:00 migration/2
     26 ?        00:00:00 ksoftirqd/2
     28 ?        00:00:00 kworker/2:0H-events_highpri
     29 ?        00:00:00 cpuhp/3
     30 ?        00:00:00 migration/3
     31 ?        00:00:00 ksoftirqd/3
     33 ?        00:00:00 kworker/3:0H-events_highpri
     34 ?        00:00:00 cpuhp/4
:
```



```
┌──(buvanesh㉿kali)-[~/demo]
└─$ ps -e | less

┌──(buvanesh㉿kali)-[~/demo]
└─$ ps -e > process_info

┌──(buvanesh㉿kali)-[~/demo]
└─$ cat process_info
    PID TTY          TIME CMD
      1 ?        00:00:01 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_kthread
     12 ?        00:00:00 rcu_tasks_rude_kthread
     13 ?        00:00:00 rcu_tasks_trace_kthread
     14 ?        00:00:00 ksoftirqd/0
     15 ?        00:00:01 rcu_preempt
     16 ?        00:00:00 migration/0
     18 ?        00:00:00 cpuhp/0
     19 ?        00:00:00 cpuhp/1
     20 ?        00:00:00 migration/1
```