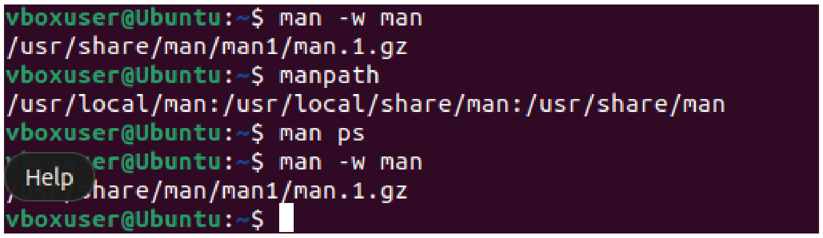
MODULE 1

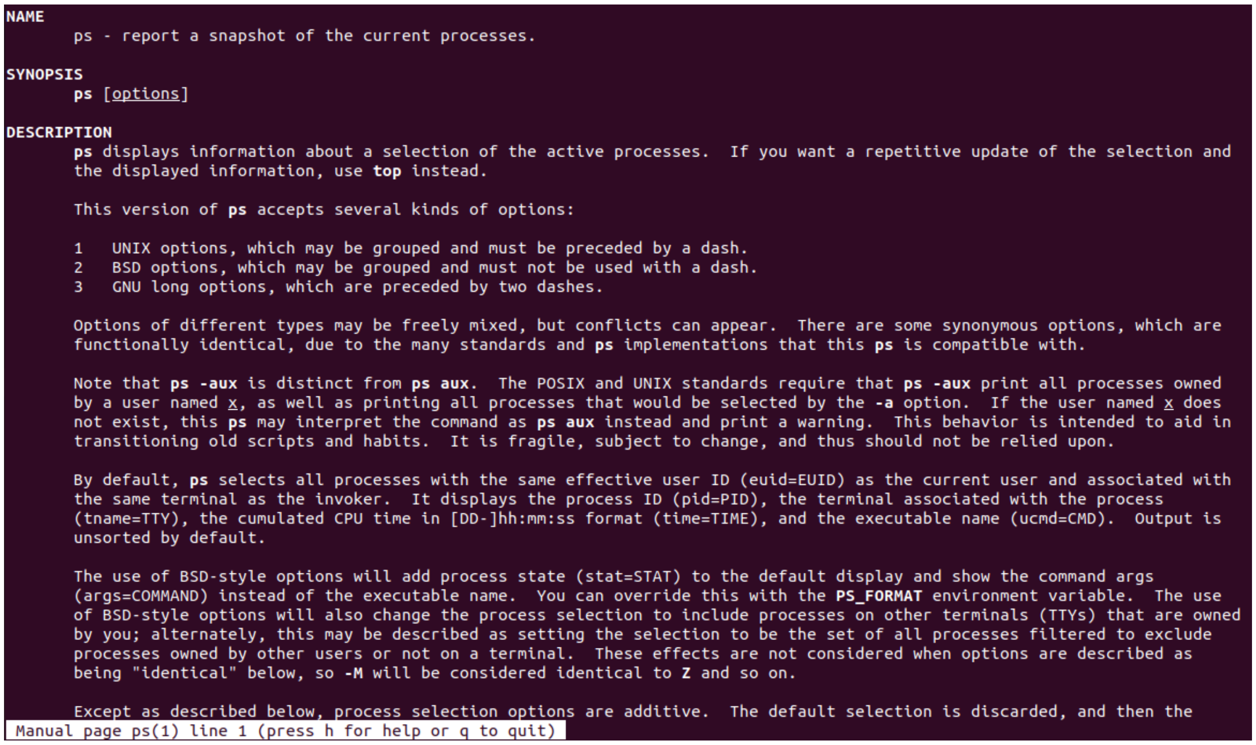
ASSIGNMENT -1

Q1. 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?



```
vboxuser@Ubuntu:~$ man -w man
/usr/share/man/man1/man.1.gz
vboxuser@Ubuntu:~$ manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
vboxuser@Ubuntu:~$ man ps
      er@Ubuntu:~$ man -w man
      hare/man/man1/man.1.gz
vboxuser@Ubuntu:~$ 
```

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

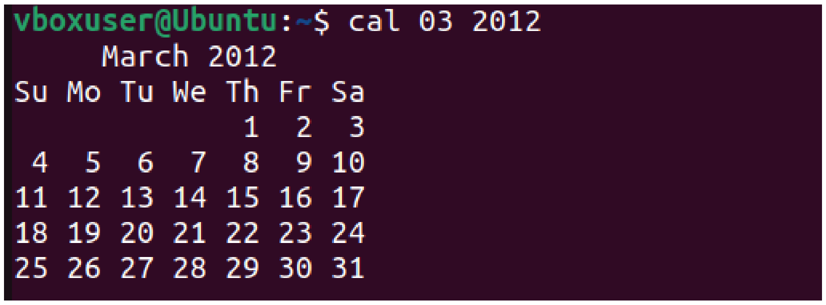


```
NAME
       ps - report a snapshot of the current processes.

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

MODULE 1

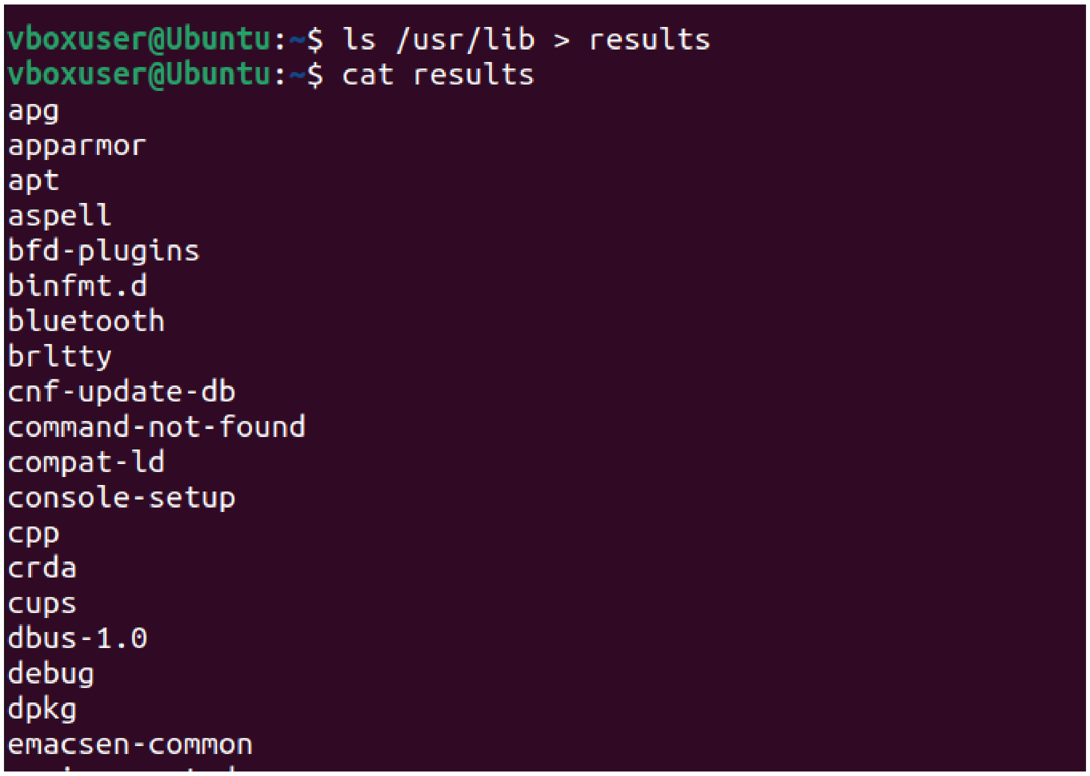
ASSIGNMENT -2

1)Display the calender for the month of March 2012



```
vboxuser@Ubuntu:~$ cal 03 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
```

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.

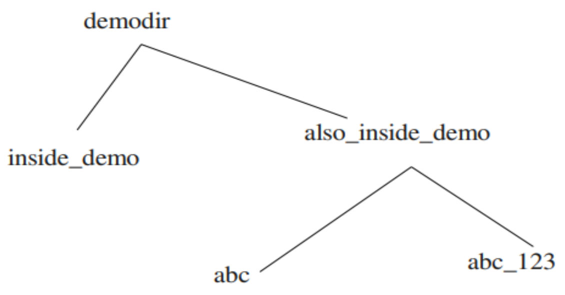


```
vboxuser@Ubuntu:~$ ls /usr/lib > results
vboxuser@Ubuntu:~$ cat results
apg
apparmor
apt
aspell
bfd-plugins
binfmt.d
bluetooth
brltty
cnf-update-db
command-not-found
compat-ld
console-setup
cpp
crda
cups
dbus-1.0
debug
dpkg
emacsen-common
```

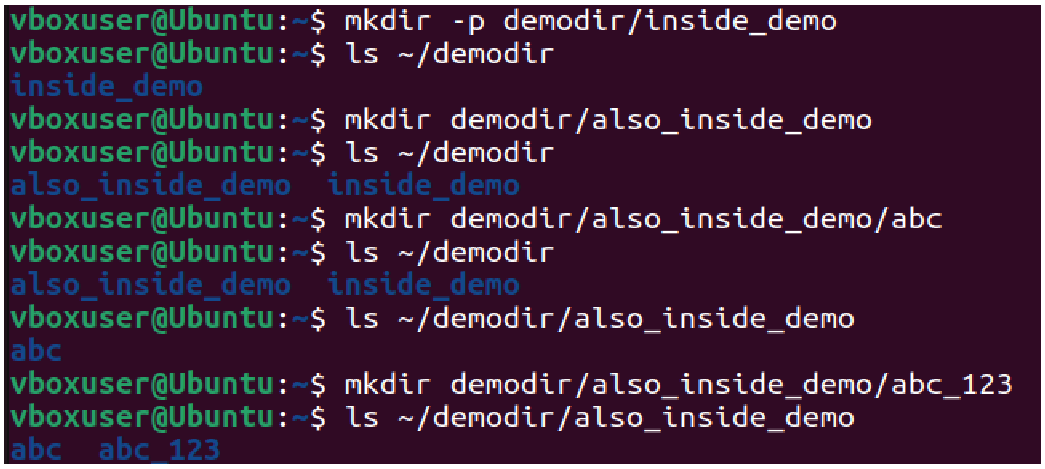
MODULE 1

ASSIGNMENT -3

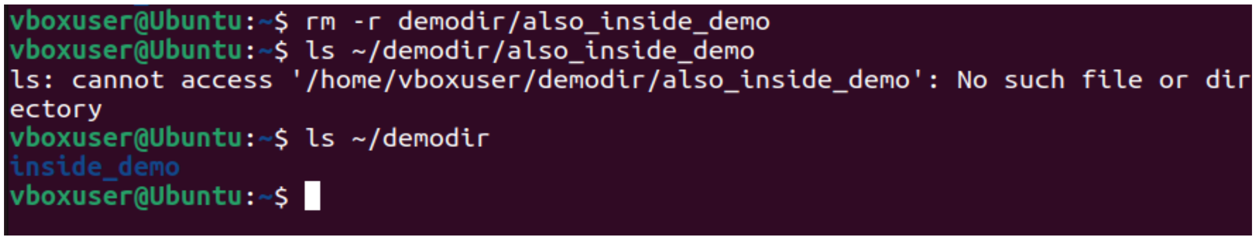
1)Make a directory structure like this in your home directory



2) Remove the also_inside_demo directory



```
vboxuser@Ubuntu:~$ mkdir -p demodir/inside_demo
vboxuser@Ubuntu:~$ ls ~/demodir
inside_demo
vboxuser@Ubuntu:~$ mkdir demodir/also_inside_demo
vboxuser@Ubuntu:~$ ls ~/demodir
also_inside_demo  inside_demo
vboxuser@Ubuntu:~$ mkdir demodir/also_inside_demo/abc
vboxuser@Ubuntu:~$ ls ~/demodir
also_inside_demo  inside_demo
vboxuser@Ubuntu:~$ ls ~/demodir/also_inside_demo
abc
vboxuser@Ubuntu:~$ mkdir demodir/also_inside_demo/abc_123
vboxuser@Ubuntu:~$ ls ~/demodir/also_inside_demo
abc  abc_123
```



```
vboxuser@Ubuntu:~$ rm -r demodir/also_inside_demo
vboxuser@Ubuntu:~$ ls ~/demodir/also_inside_demo
ls: cannot access '/home/vboxuser/demodir/also_inside_demo': No such file or dir
ectory
vboxuser@Ubuntu:~$ ls ~/demodir
inside_demo
vboxuser@Ubuntu:~$ 
```

MODULE 1

ASSIGNMENT -4

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

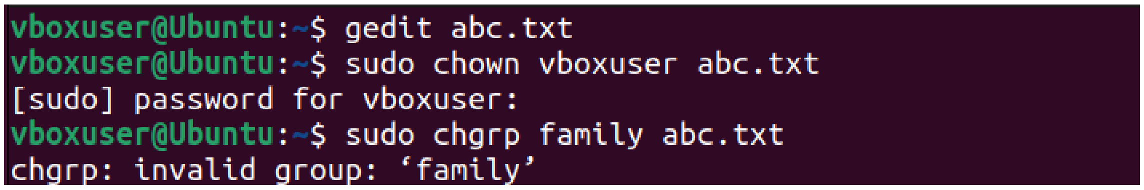
   and also change the group to family.

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

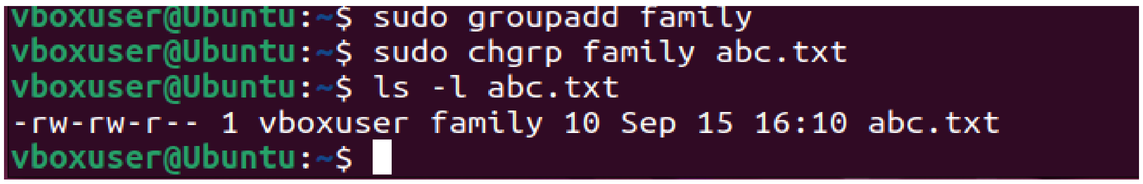
3. Create a file test.txt on your desktop and identify its inode number, also create a softlink

   for test.txt in your home.

1.

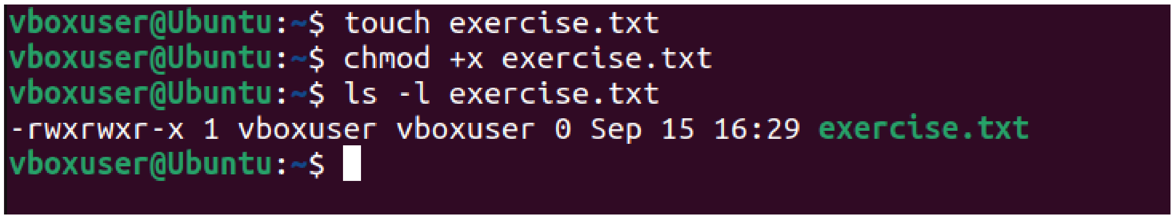


```
vboxuser@Ubuntu:~$ gedit abc.txt
vboxuser@Ubuntu:~$ sudo chown vboxuser abc.txt
[sudo] password for vboxuser:
vboxuser@Ubuntu:~$ sudo chgrp family abc.txt
chgrp: invalid group: ‘family’
```



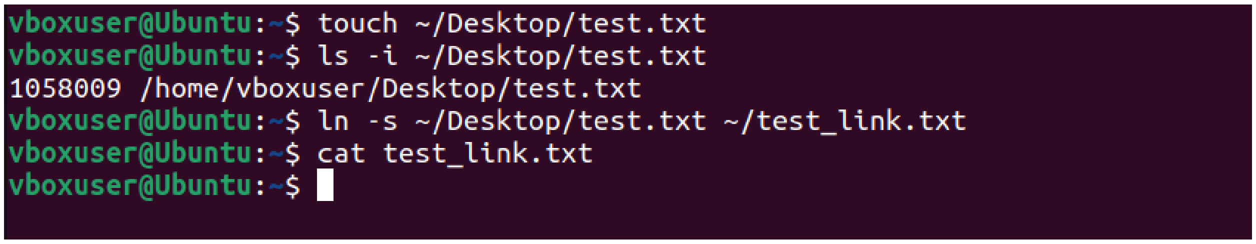
```
vboxuser@Ubuntu:~$ sudo groupadd family
vboxuser@Ubuntu:~$ sudo chgrp family abc.txt
vboxuser@Ubuntu:~$ ls -l abc.txt
-rw-rw-r-- 1 vboxuser family 10 Sep 15 16:10 abc.txt
vboxuser@Ubuntu:~$ 
```

2.



```
vboxuser@Ubuntu:~$ touch exercise.txt
vboxuser@Ubuntu:~$ chmod +x exercise.txt
vboxuser@Ubuntu:~$ ls -l exercise.txt
-rwxrwxr-x 1 vboxuser vboxuser 0 Sep 15 16:29 exercise.txt
vboxuser@Ubuntu:~$ 
```

3.



```
vboxuser@Ubuntu:~$ touch ~/Desktop/test.txt
vboxuser@Ubuntu:~$ ls -i ~/Desktop/test.txt
1058009 /home/vboxuser/Desktop/test.txt
vboxuser@Ubuntu:~$ ln -s ~/Desktop/test.txt ~/test_link.txt
vboxuser@Ubuntu:~$ cat test_link.txt
vboxuser@Ubuntu:~$ 
```

MODULE 1

ASSIGNMENT -5

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

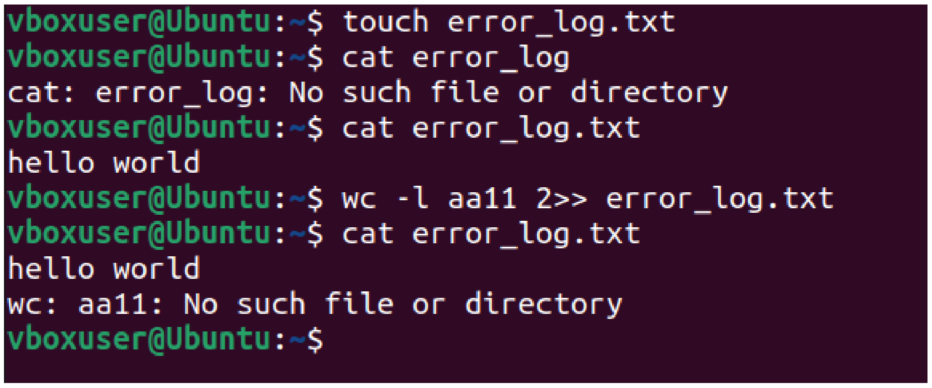
How can you redirect the error message to the file error_log when we apply the command "wc -l aa11" ?

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

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 ?

1.



```
vboxuser@Ubuntu:~$ touch error_log.txt
vboxuser@Ubuntu:~$ cat error_log
cat: error_log: No such file or directory
vboxuser@Ubuntu:~$ cat error_log.txt
hello world
vboxuser@Ubuntu:~$ wc -l aa11 2>> error_log.txt
vboxuser@Ubuntu:~$ cat error_log.txt
hello world
wc: aa11: No such file or directory
vboxuser@Ubuntu:~$
```

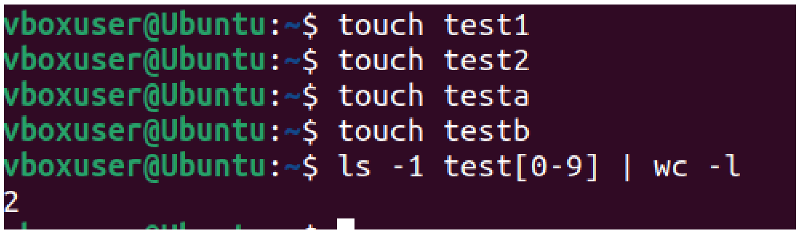
2.

ls -1 test[0-9] lists all files in the current directory that start with "test" and have a single digit (0-9) following it. The **[0-9]** is a regular expression pattern that matches any single digit.

I pipes the output of the ls command (the list of files) to the wc -l command.

wc -l counts the number of lines in the input it receives. Since each line corresponds to a file name in this case, it will count the number of matching files.

When we run this command, it will display the count of files that match the specified criteria, which are files starting with "test" and having a single digit in their name.



```
vboxuser@Ubuntu:~$ touch test1
vboxuser@Ubuntu:~$ touch test2
vboxuser@Ubuntu:~$ touch testa
vboxuser@Ubuntu:~$ touch testb
vboxuser@Ubuntu:~$ ls -1 test[0-9] | wc -l
2
```

MODULE 1

ASSIGNMENT -6

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,

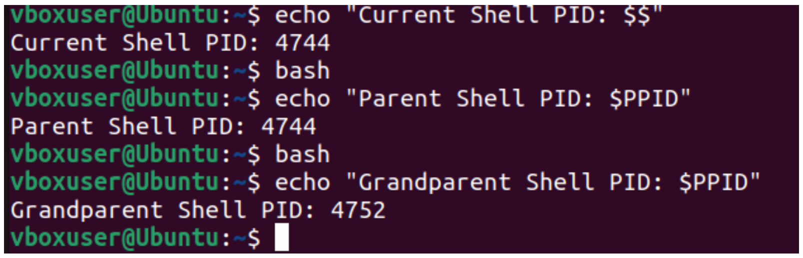
how can you see the PID of the current shell ? How can you see the PID of the shell which is the grandparent of the current shell?

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 ?

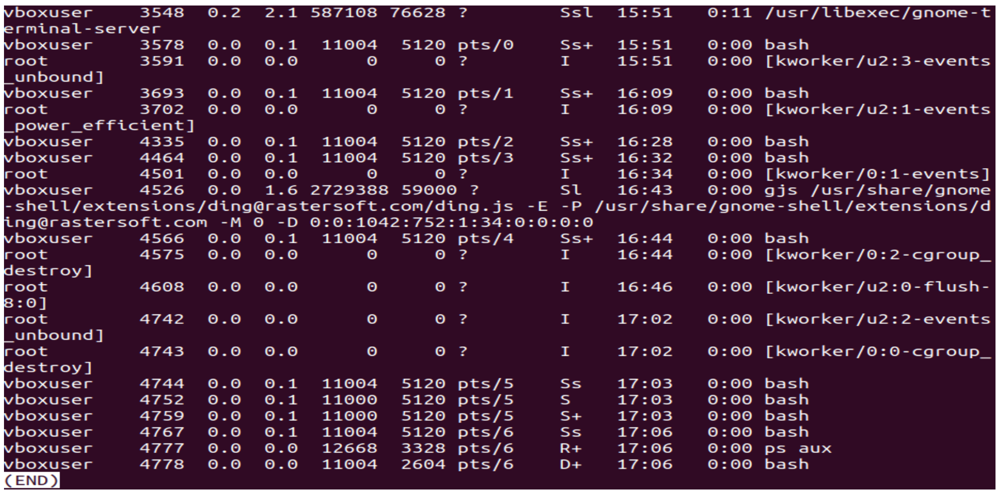
How can you store the output in a file named process_info?

1.



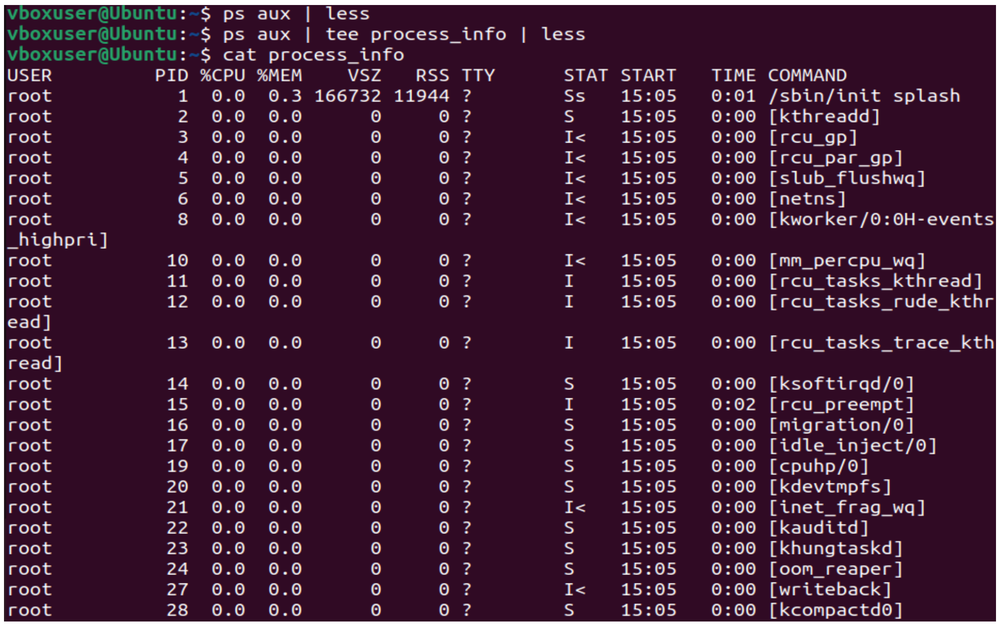
```
vboxuser@Ubuntu:~$ echo "Current Shell PID: $$"
Current Shell PID: 4744
vboxuser@Ubuntu:~$ bash
vboxuser@Ubuntu:~$ echo "Parent Shell PID: $PPID"
Parent Shell PID: 4744
vboxuser@Ubuntu:~$ bash
vboxuser@Ubuntu:~$ echo "Grandparent Shell PID: $PPID"
Grandparent Shell PID: 4752
vboxuser@Ubuntu:~$ 
```

2. Using ps aux| less command, we can view the processes that are currently running on the computer and scroll through them.



```
vboxuser     3548  0.2  2.1 587108 76628 ?        Ssl  15:51   0:11 /usr/libexec/gnome-t
erminal-server
vboxuser     3578  0.0  0.1  11004  5120 pts/0    Ss+  15:51   0:00 bash
root         3591  0.0  0.0      0     0 ?        I    15:51   0:00 [kworker/u2:3-events
_unbound]
vboxuser     3693  0.0  0.1  11004  5120 pts/1    Ss+  16:09   0:00 bash
root         3702  0.0  0.0      0     0 ?        I    16:09   0:00 [kworker/u2:1-events
_power_efficient]
vboxuser     4335  0.0  0.1  11004  5120 pts/2    Ss+  16:28   0:00 bash
vboxuser     4464  0.0  0.1  11004  5120 pts/3    Ss+  16:32   0:00 bash
root         4501  0.0  0.0      0     0 ?        I    16:34   0:00 [kworker/0:1-events]
vboxuser     4526  0.0  1.6 2729388 59000 ?      Sl   16:43   0:00 gjs /usr/share/gnome
-shell/extensions/ding@rastersoft.com/ding.js -E -P /usr/share/gnome-shell/extensions/d
ing@rastersoft.com -M 0 -D 0:0:1042:752:1:34:0:0:0:0
vboxuser     4566  0.0  0.1  11004  5120 pts/4    Ss+  16:44   0:00 bash
root         4575  0.0  0.0      0     0 ?        I    16:44   0:00 [kworker/0:2-cgroup_
destroy]
root         4608  0.0  0.0      0     0 ?        I    16:46   0:00 [kworker/u2:0-flush-
8:0]
root         4742  0.0  0.0      0     0 ?        I    17:02   0:00 [kworker/u2:2-events
_unbound]
root         4743  0.0  0.0      0     0 ?        I    17:02   0:00 [kworker/0:0-cgroup_
destroy]
vboxuser     4744  0.0  0.1  11004  5120 pts/5    Ss   17:03   0:00 bash
vboxuser     4752  0.0  0.1  11000  5120 pts/5    S    17:03   0:00 bash
vboxuser     4759  0.0  0.1  11000  5120 pts/5    S+   17:03   0:00 bash
vboxuser     4767  0.0  0.1  11004  5120 pts/6    Ss   17:06   0:00 bash
vboxuser     4777  0.0  0.0  12668  3328 pts/6    R+   17:06   0:00 ps aux
vboxuser     4778  0.0  0.0  11004  2604 pts/6    D+   17:06   0:00 bash
(END)
```

We can input the processes into process_info by using tee command



```
vboxuser@Ubuntu:~$ ps aux | less
vboxuser@Ubuntu:~$ ps aux | tee process_info | less
vboxuser@Ubuntu:~$ cat process_info
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 166732 11944 ?        Ss   15:05   0:01 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    15:05   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   15:05   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   15:05   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   15:05   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   15:05   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   15:05   0:00 [kworker/0:0H-events
_highpri]
root          10  0.0  0.0      0     0 ?        I<   15:05   0:00 [mm_percpu_wq]
root          11  0.0  0.0      0     0 ?        I    15:05   0:00 [rcu_tasks_kthread]
root          12  0.0  0.0      0     0 ?        I    15:05   0:00 [rcu_tasks_rude_kthr
ead]
root          13  0.0  0.0      0     0 ?        I    15:05   0:00 [rcu_tasks_trace_kth
read]
root          14  0.0  0.0      0     0 ?        S    15:05   0:00 [ksoftirqd/0]
root          15  0.0  0.0      0     0 ?        I    15:05   0:02 [rcu_preempt]
root          16  0.0  0.0      0     0 ?        S    15:05   0:00 [migration/0]
root          17  0.0  0.0      0     0 ?        S    15:05   0:00 [idle_inject/0]
root          19  0.0  0.0      0     0 ?        S    15:05   0:00 [cpuhp/0]
root          20  0.0  0.0      0     0 ?        S    15:05   0:00 [kdevtmpfs]
root          21  0.0  0.0      0     0 ?        I<   15:05   0:00 [inet_frag_wq]
root          22  0.0  0.0      0     0 ?        S    15:05   0:00 [kauditd]
root          23  0.0  0.0      0     0 ?        S    15:05   0:00 [khungtaskd]
root          24  0.0  0.0      0     0 ?        S    15:05   0:00 [oom_reaper]
root          27  0.0  0.0      0     0 ?        I<   15:05   0:00 [writeback]
root          28  0.0  0.0      0     0 ?        S    15:05   0:00 [kcompactd0]
```