Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 06
Course code: ICT-3108
Course title: Operating system   Lab
Date of Performance:11-09-20
Date of Submission: 12-09020

Submitted by

Name: Md.Abdullah Al mamun
ID:IT-18040
3th year 1ndsemester
Session: 2017-2018
Dept. of ICT
MBSTU.

Submitted To

Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

Lab Report No : 06

Lab Report Name : Linux command for process

Theory

1. How to Manage Processes from the Linux Terminal?

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

Linux Processes Management

1. Starting a Process. When you start a process(run a command), there are two ways you can run it – ...
2. Listing Running Processes. ...
3. Stopping Processes. ...
4. Parent and Child Processes. ...
5. Zombie and Orphan Processes. ...
6. Daemon Processes. ...
7. The top Command. ...
8. Job ID Versus Process ID.

Starting a Process
When we start a process (run a command), there are two ways we can

1.Foreground Processes

2 .background processes

By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.

You can see this happen with the **ls** command. If you wish to list all the files in your current directory, you can use the following command –

```
$ls ch*.doc
```

This would display all the files, the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

The process runs in the foreground, the output is directed to my screen, and if the **ls** command wants any input (which it does not), it waits for it from the keyboard.

While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.

Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (**&**) at the end of the command.

```
$ls ch*.doc &
```

This displays all those files the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

Here, if the **ls** command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard.

That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground.

Press the Enter key and you will see the following –

```
[1]   +   Done              ls ch*.doc &
$
```

The first line tells you that the **ls** command background process finishes successfully. The second is a prompt for another command.

Listing Running Processes

It is easy to see your own processes by running the **ps** (process status) command as follows –

```
$ps
PID       TTY      TIME        CMD
18358    ttyp3    00:00:00    sh
18361    ttyp3    00:01:31    abiword
18789    ttyp3    00:00:00    ps
```

One of the most commonly used flags for ps is the **-f** (f for full) option, which provides more information as shown in the following example –

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

Here is the description of all the fields displayed by **ps -f** command –

Sr.No.	**Column & Description**
1	**UID** User ID that this process belongs to (the person running it)
2	**PID** Process ID
3	**PPID**

	Parent process ID (the ID of the process that started it)
4	**C** CPU utilization of process
5	**STIME** Process start time
6	**TTY** Terminal type associated with the process
7	**TIME** CPU time taken by the process
8	**CMD** The command that started this process

There are other options which can be used along with **ps** command –

Sr.No.	Option & Description
1	**-a** Shows information about all users
2	**-x** Shows information about processes without terminals

3	**-u** Shows additional information like -f option
4	**-e** Displays extended information

Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows –

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

Here, the **kill** command terminates the **first_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows –

```
$kill -9 6738
Terminated
```

Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

Daemon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot ope /dev/tty. If you do a "ps -ef" and look at the tty field, all daemons will have a ? for the tty.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then it’s worth to make it a daemon and run it in the background.

The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

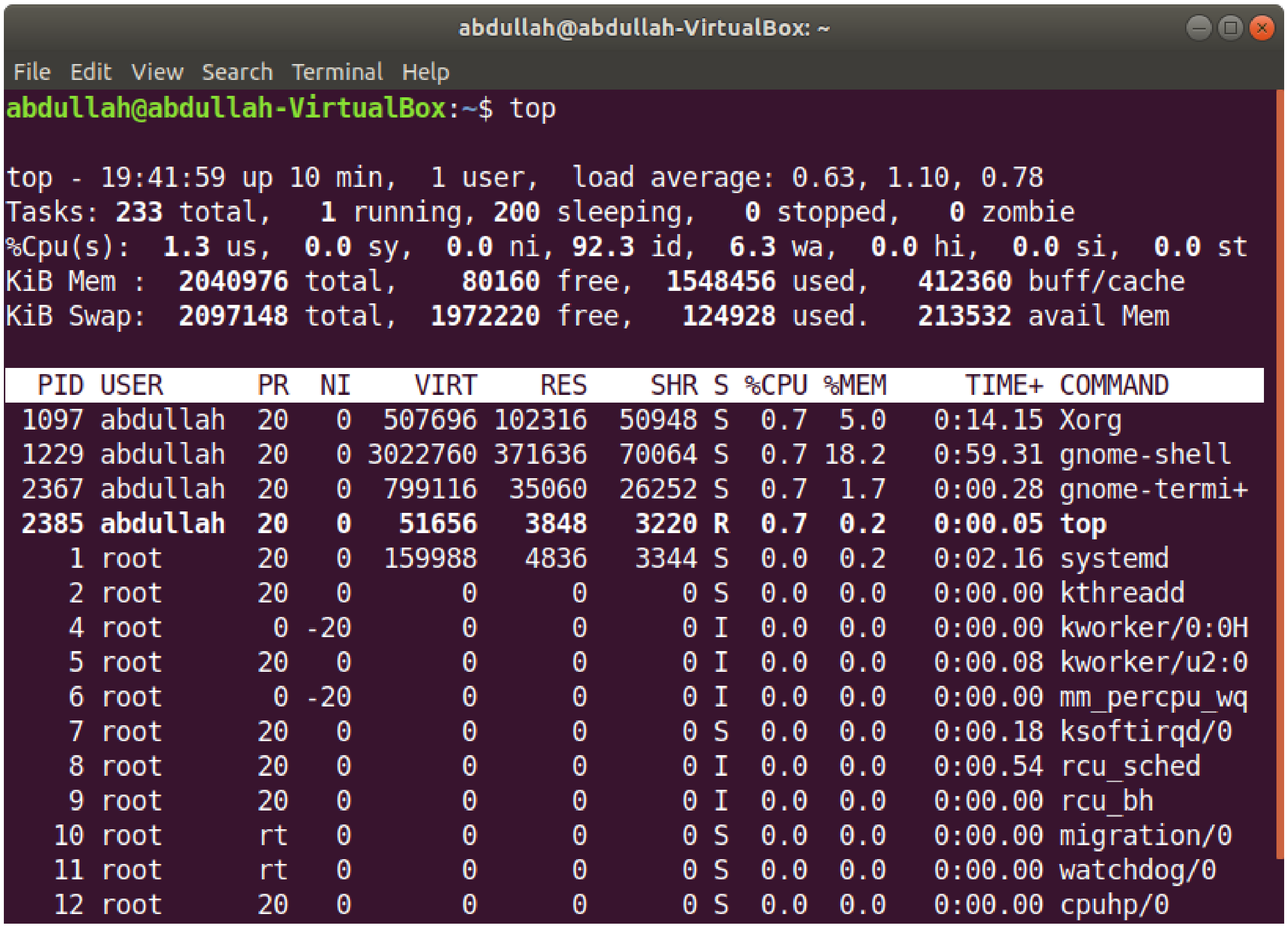
Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes –

```
$top
```

2 . Run the following process commands in Linux.

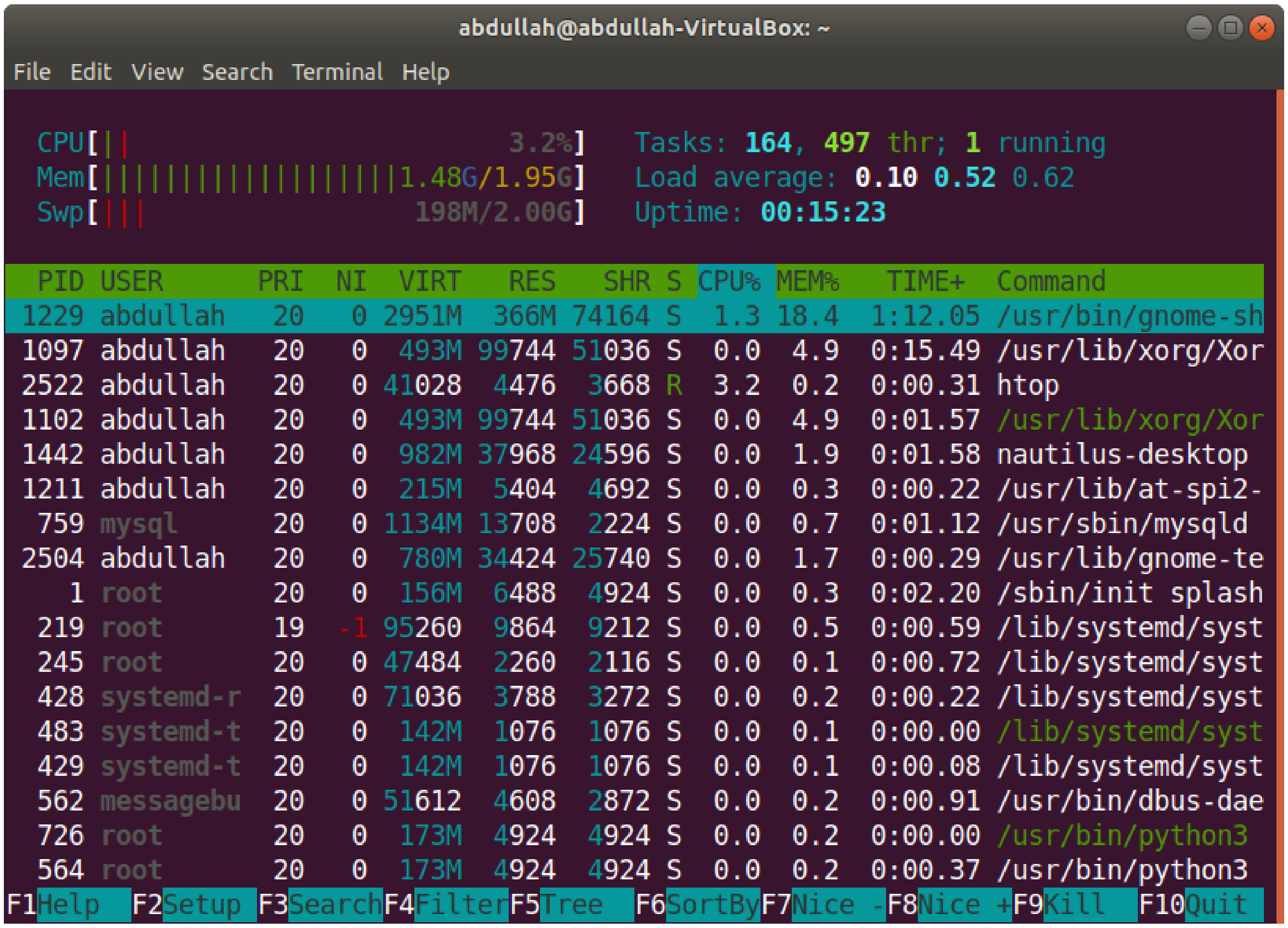
Top, htop, Ps, pstree, kill, pgrep, pkill ,killall, renice, xkill.

a) Top


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

```
abdullah@abdullah-VirtualBox:~$ top

top - 19:41:59 up 10 min,  1 user,  load average: 0.63, 1.10, 0.78
Tasks: 233 total,   1 running, 200 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.3 us,  0.0 sy,  0.0 ni, 92.3 id,  6.3 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2040976 total,    80160 free,  1548456 used,   412360 buff/cache
KiB Swap:  2097148 total,  1972220 free,   124928 used.   213532 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1097 abdullah  20   0  507696 102316  50948 S  0.7  5.0   0:14.15 Xorg
 1229 abdullah  20   0 3022760 371636  70064 S  0.7 18.2   0:59.31 gnome-shell
 2367 abdullah  20   0  799116  35060  26252 S  0.7  1.7   0:00.28 gnome-termi+
 2385 abdullah  20   0   51656   3848   3220 R  0.7  0.2   0:00.05 top
    1 root      20   0  159988   4836   3344 S  0.0  0.2   0:02.16 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H
    5 root      20   0       0      0      0 I  0.0  0.0   0:00.08 kworker/u2:0
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.18 ksoftirqd/0
    8 root      20   0       0      0      0 I  0.0  0.0   0:00.54 rcu_sched
    9 root      20   0       0      0      0 I  0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 watchdog/0
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 cpuhp/0
```

b) htop

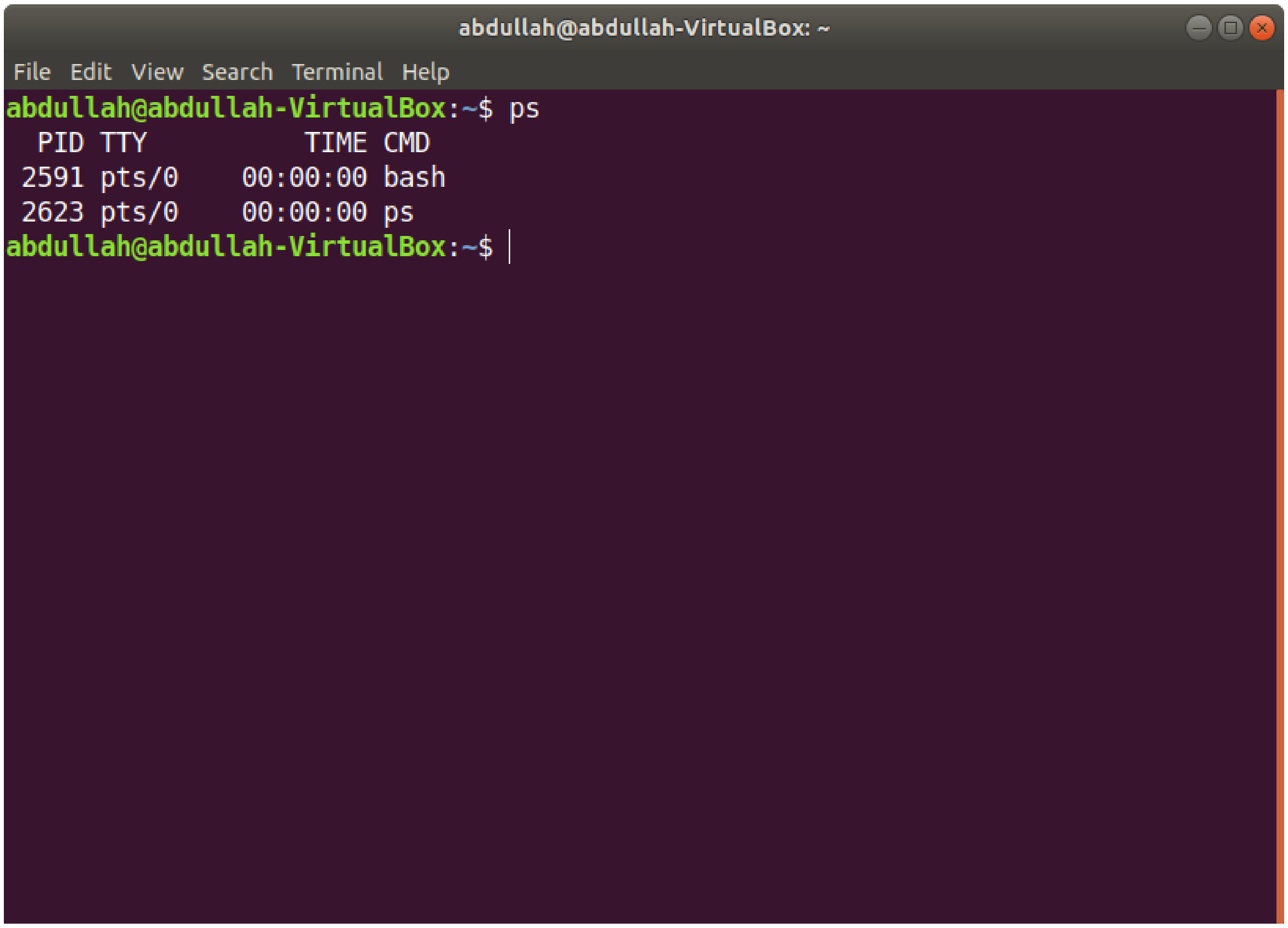


abdullah@abdullah-VirtualBox: ~

File Edit View Search Terminal Help

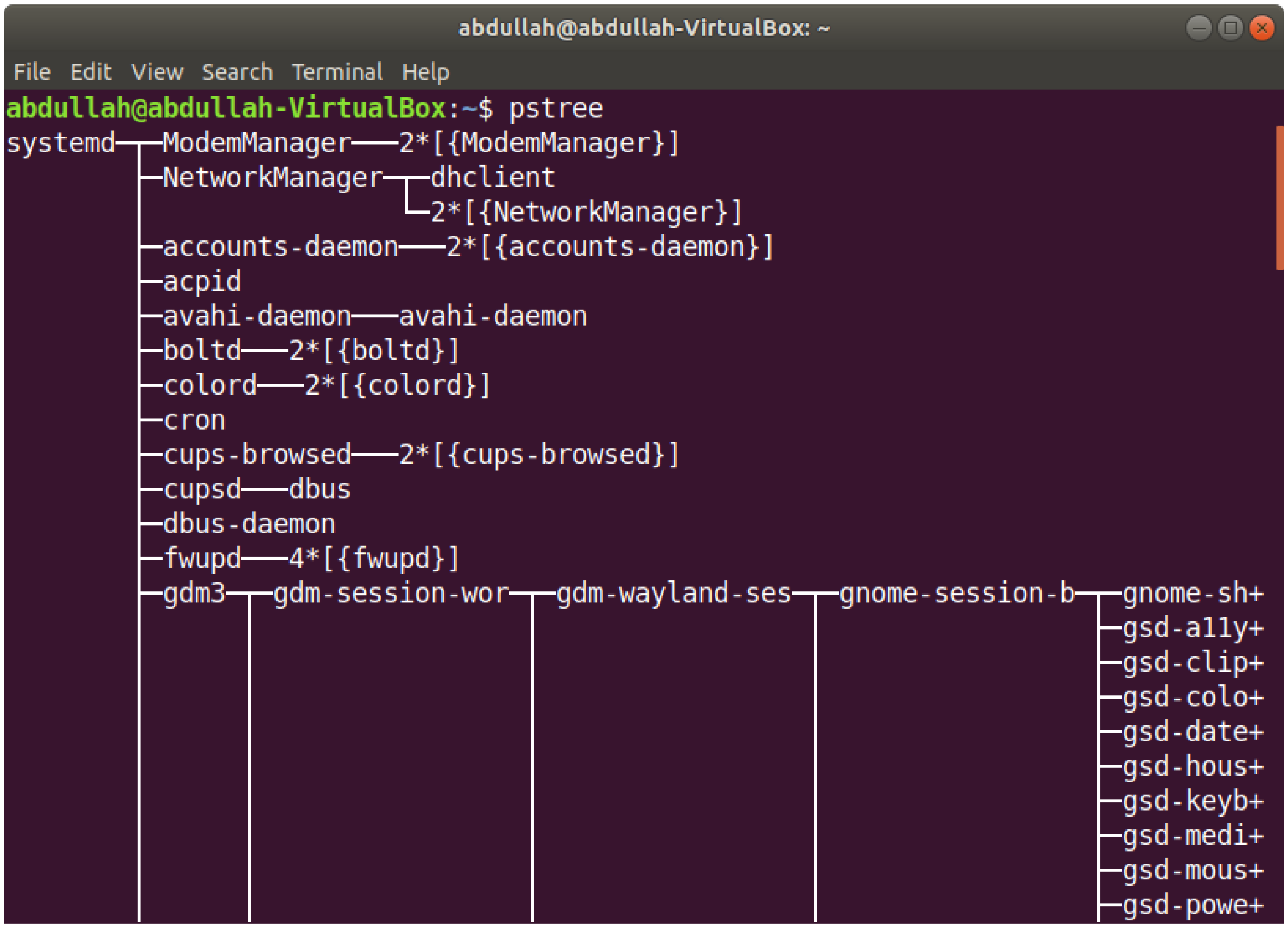
```
  CPU[||                         3.2%]   Tasks: 164, 497 thr; 1 running
  Mem[||||||||||||||||||1.48G/1.95G]   Load average: 0.10 0.52 0.62
  Swp[|||                  198M/2.00G]   Uptime: 00:15:23

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1229 abdullah   20   0 2951M  366M 74164 S  1.3 18.4  1:12.05 /usr/bin/gnome-sh
 1097 abdullah   20   0  493M 99744 51036 S  0.0  4.9  0:15.49 /usr/lib/xorg/Xor
 2522 abdullah   20   0 41028  4476  3668 R  3.2  0.2  0:00.31 htop
 1102 abdullah   20   0  493M 99744 51036 S  0.0  4.9  0:01.57 /usr/lib/xorg/Xor
 1442 abdullah   20   0  982M 37968 24596 S  0.0  1.9  0:01.58 nautilus-desktop
 1211 abdullah   20   0  215M  5404  4692 S  0.0  0.3  0:00.22 /usr/lib/at-spi2-
  759 mysql      20   0 1134M 13708  2224 S  0.0  0.7  0:01.12 /usr/sbin/mysqld
 2504 abdullah   20   0  780M 34424 25740 S  0.0  1.7  0:00.29 /usr/lib/gnome-te
    1 root       20   0  156M  6488  4924 S  0.0  0.3  0:02.20 /sbin/init splash
  219 root       19  -1 95260  9864  9212 S  0.0  0.5  0:00.59 /lib/systemd/syst
  245 root       20   0 47484  2260  2116 S  0.0  0.1  0:00.72 /lib/systemd/syst
  428 systemd-r  20   0 71036  3788  3272 S  0.0  0.2  0:00.22 /lib/systemd/syst
  483 systemd-t  20   0  142M  1076  1076 S  0.0  0.1  0:00.00 /lib/systemd/syst
  429 systemd-t  20   0  142M  1076  1076 S  0.0  0.1  0:00.08 /lib/systemd/syst
  562 messagebu  20   0 51612  4608  2872 S  0.0  0.2  0:00.91 /usr/bin/dbus-dae
  726 root       20   0  173M  4924  4924 S  0.0  0.2  0:00.00 /usr/bin/python3
  564 root       20   0  173M  4924  4924 S  0.0  0.2  0:00.37 /usr/bin/python3
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

c) ps


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

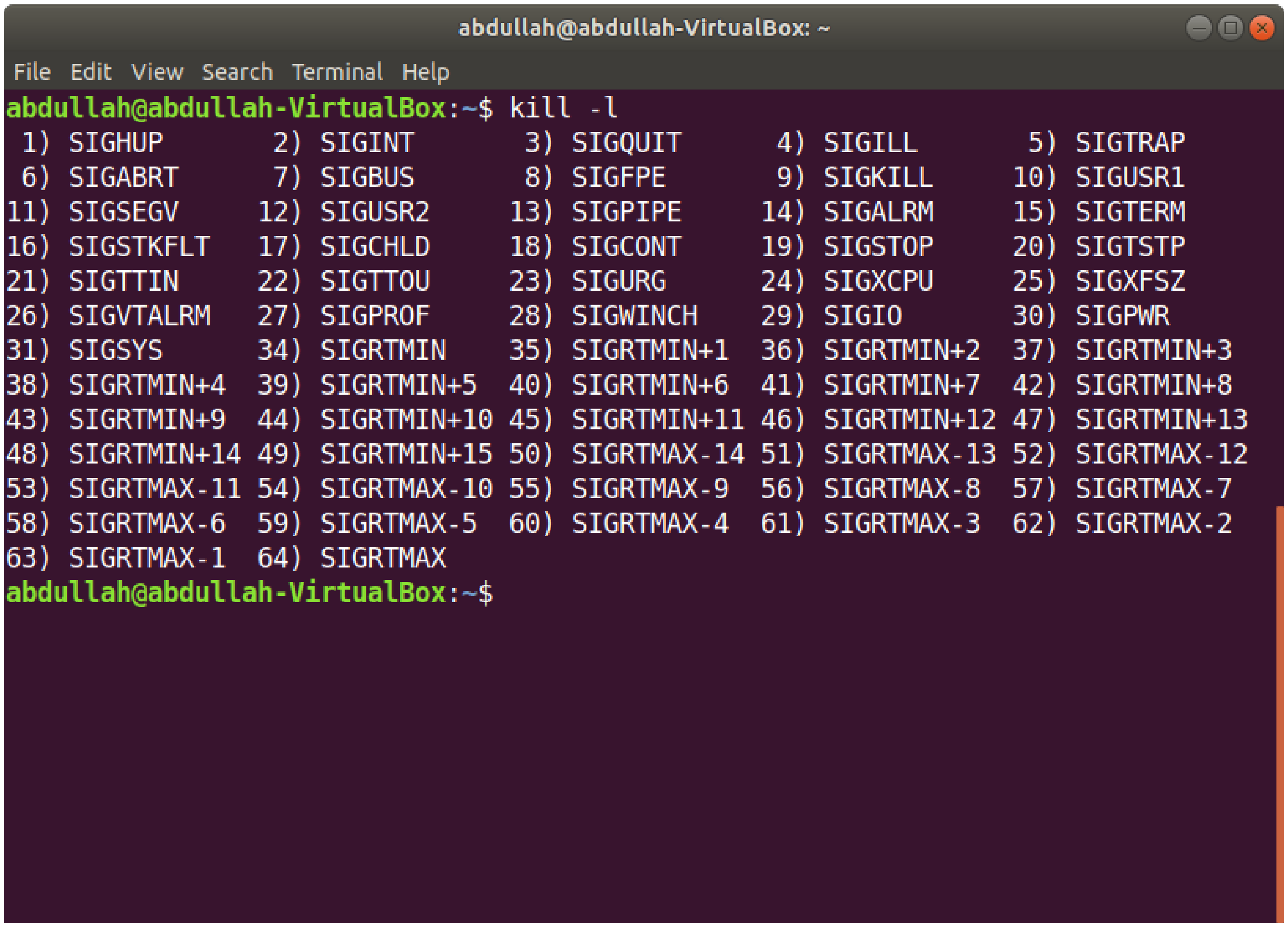
```
abdullah@abdullah-VirtualBox:~$ ps
  PID TTY          TIME CMD
 2591 pts/0    00:00:00 bash
 2623 pts/0    00:00:00 ps
abdullah@abdullah-VirtualBox:~$
```

d) pstree


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

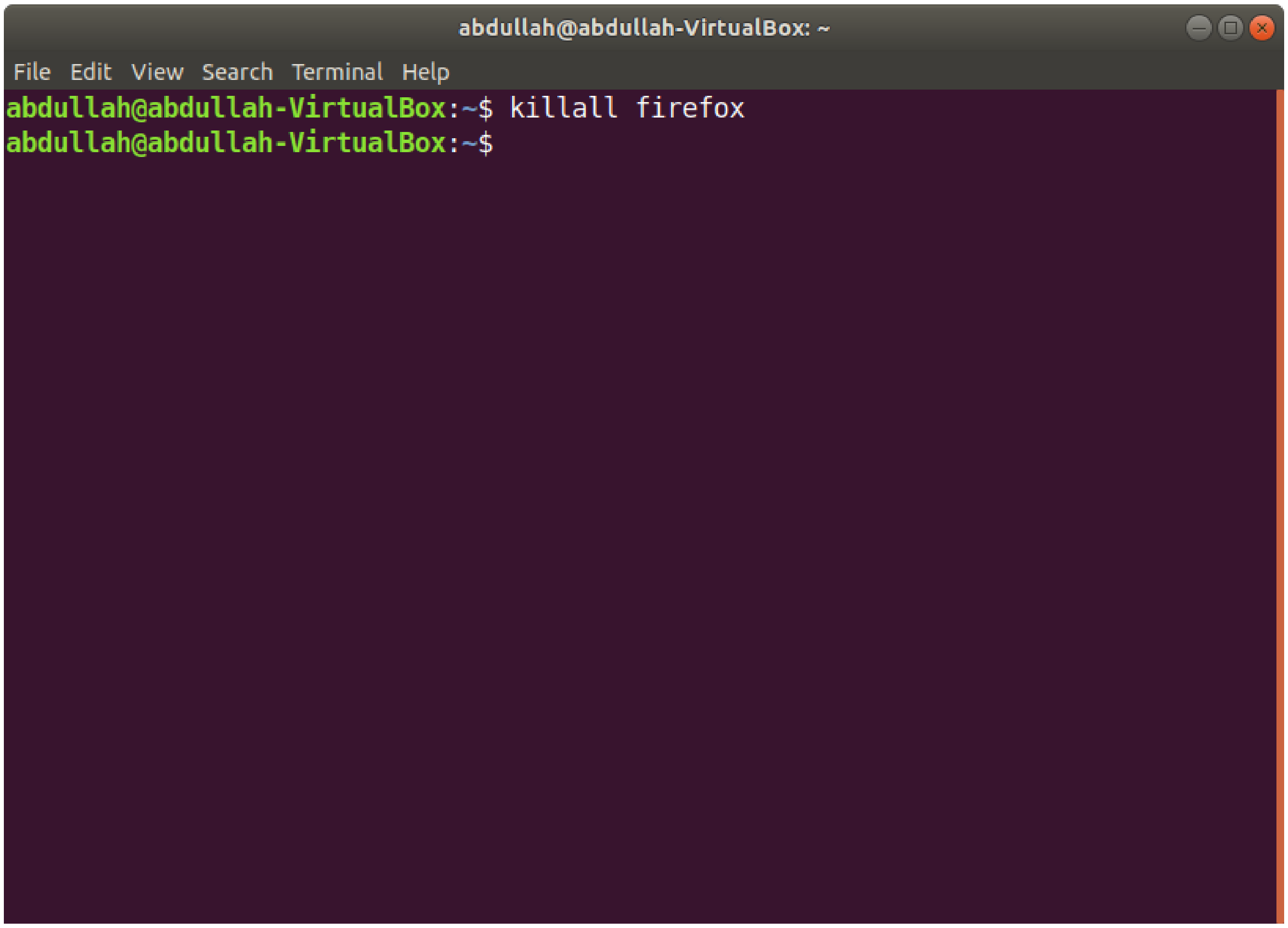
```
abdullah@abdullah-VirtualBox:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─boltd───2*[{boltd}]
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───dbus
        ├─dbus-daemon
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b─┬─gnome-sh+
        │      │                 │                 │                 ├─gsd-a11y+
        │      │                 │                 │                 ├─gsd-clip+
        │      │                 │                 │                 ├─gsd-colo+
        │      │                 │                 │                 ├─gsd-date+
        │      │                 │                 │                 ├─gsd-hous+
        │      │                 │                 │                 ├─gsd-keyb+
        │      │                 │                 │                 ├─gsd-medi+
        │      │                 │                 │                 ├─gsd-mous+
        │      │                 │                 │                 ├─gsd-powe+
```

e) kill -l


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

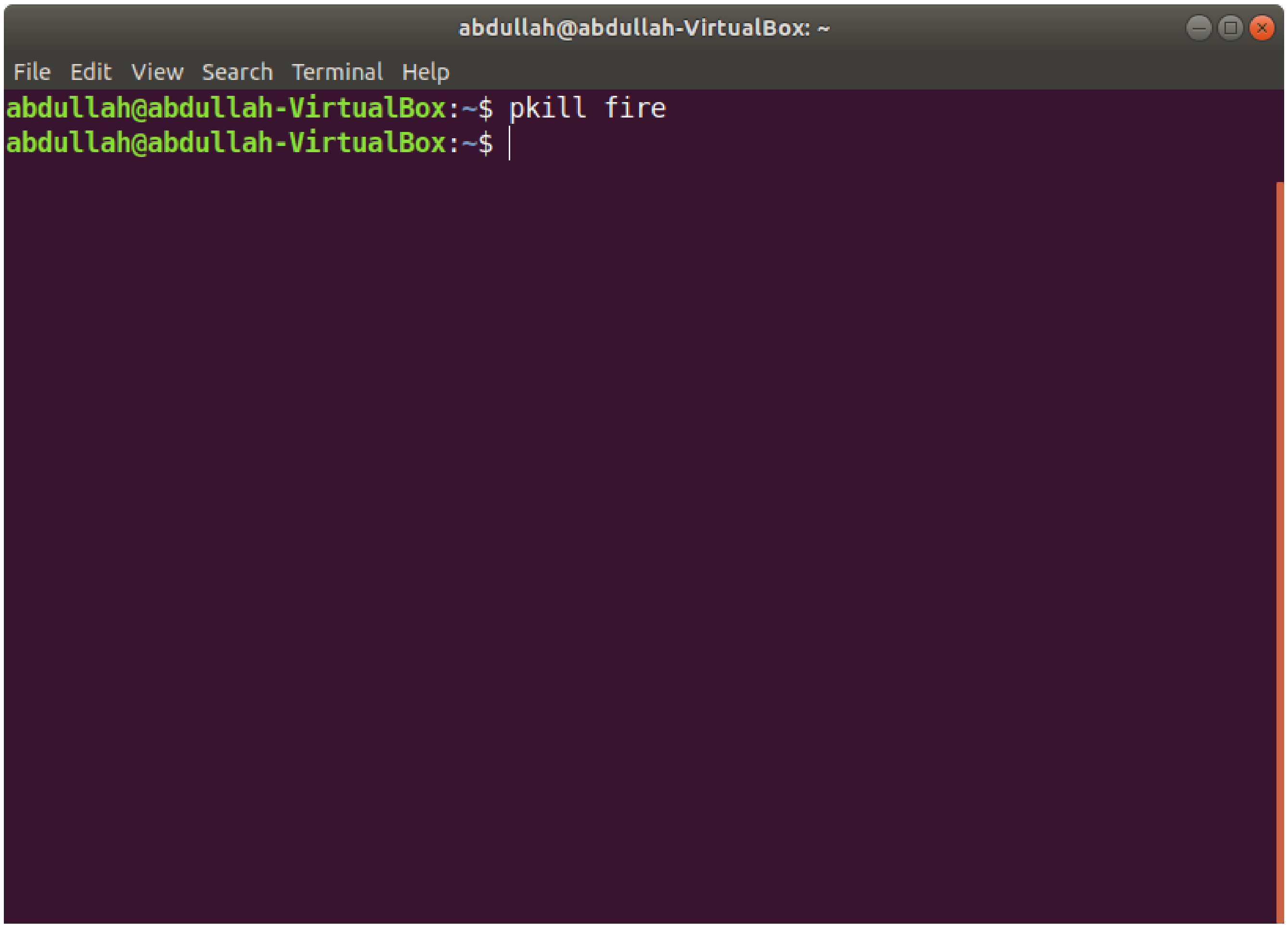
```
abdullah@abdullah-VirtualBox:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
abdullah@abdullah-VirtualBox:~$
```

f) killall


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

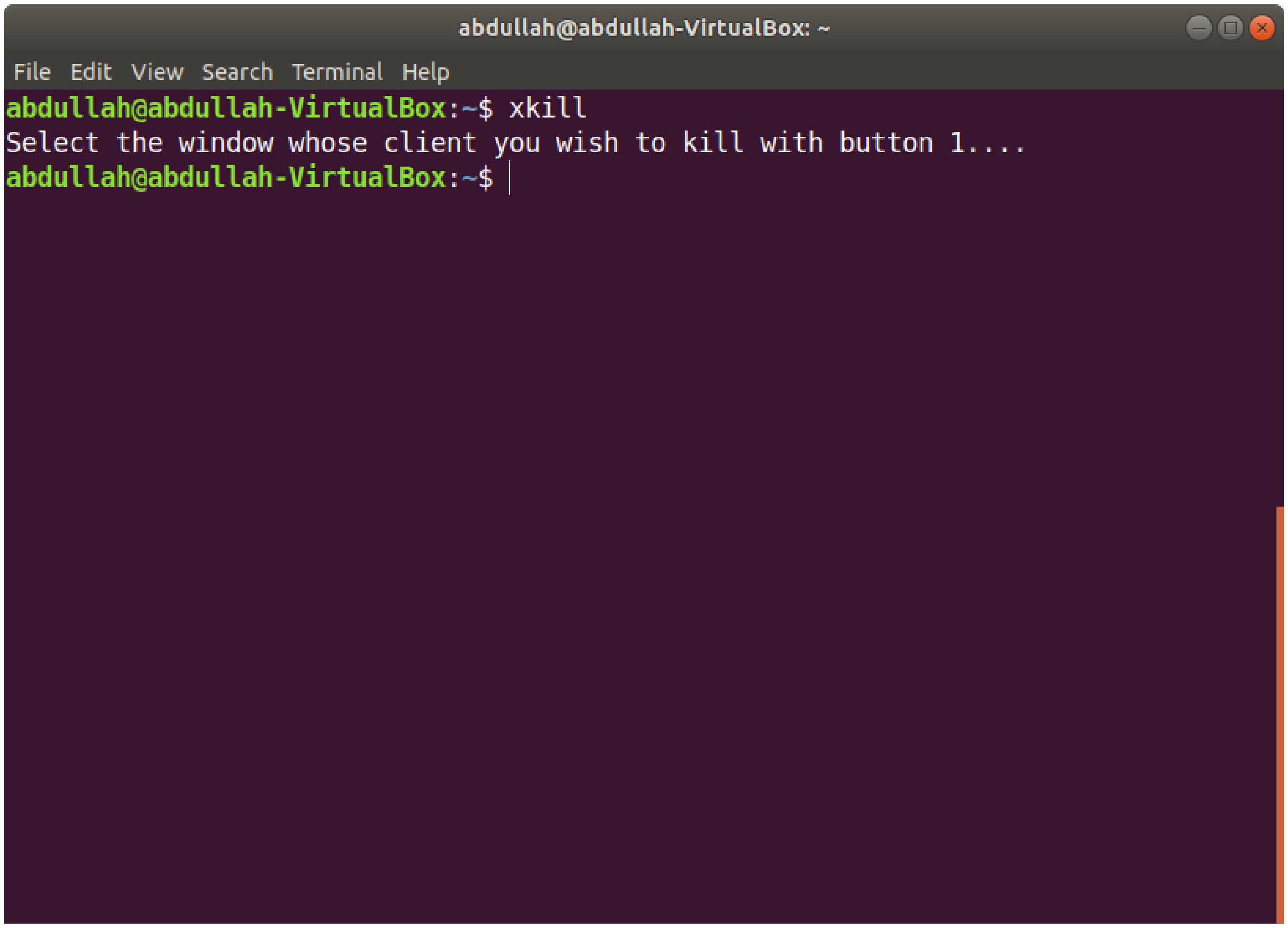
```
abdullah@abdullah-VirtualBox:~$ killall firefox
abdullah@abdullah-VirtualBox:~$
```

g) pkill


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

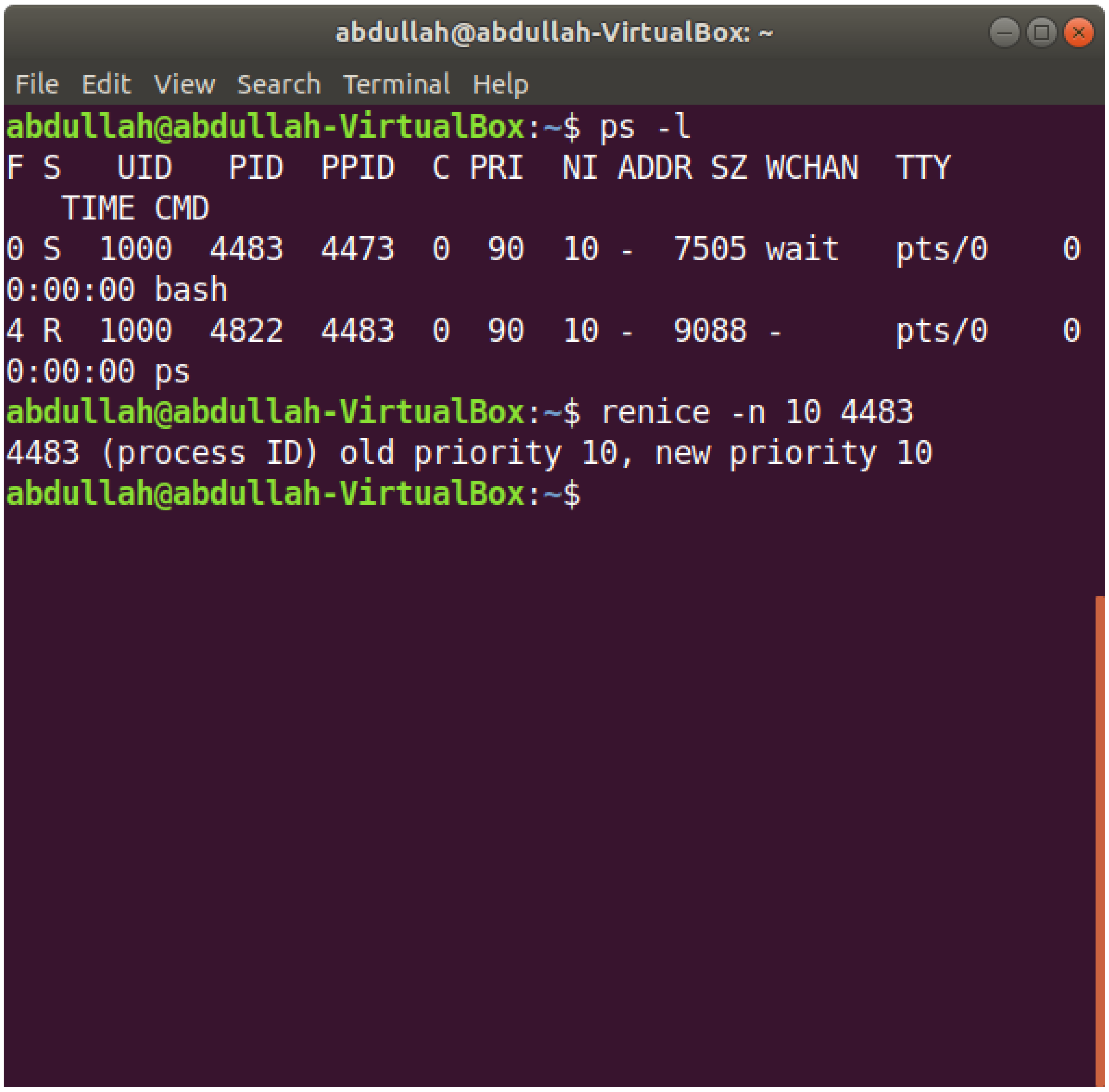
```
abdullah@abdullah-VirtualBox:~$ pkill fire
abdullah@abdullah-VirtualBox:~$ 
```

h) xkill


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

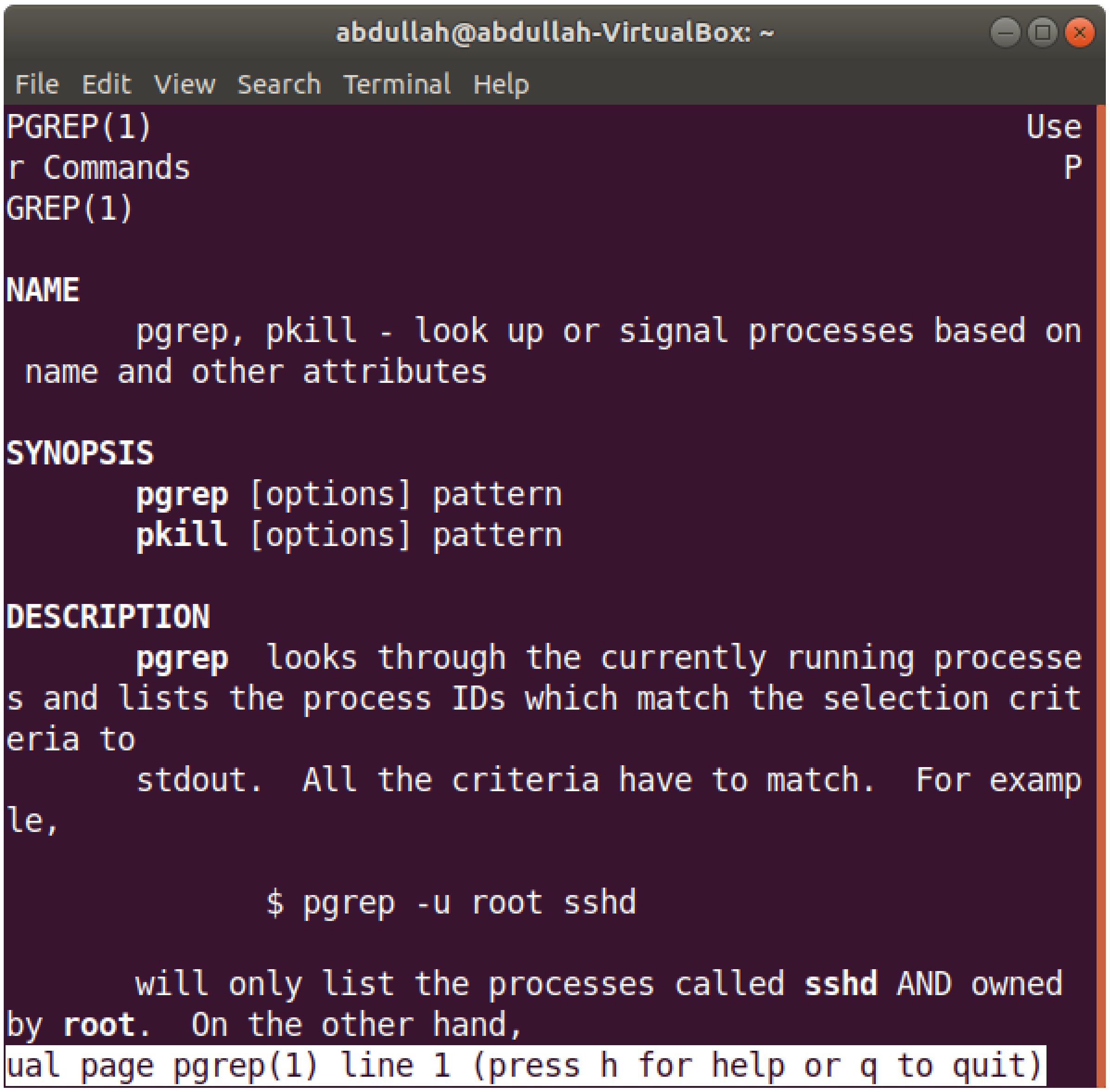
```
abdullah@abdullah-VirtualBox:~$ xkill
Select the window whose client you wish to kill with button 1....
abdullah@abdullah-VirtualBox:~$ 
```

I) ps-l and renice


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

```
abdullah@abdullah-VirtualBox:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY
   TIME CMD
0 S  1000  4483  4473  0  90  10 -  7505 wait   pts/0    0
0:00:00 bash
4 R  1000  4822  4483  0  90  10 -  9088 -      pts/0    0
0:00:00 ps
abdullah@abdullah-VirtualBox:~$ renice -n 10 4483
4483 (process ID) old priority 10, new priority 10
abdullah@abdullah-VirtualBox:~$
```

j) pgrep


abdullah@abdullah-VirtualBox: ~
File Edit View Search Terminal Help

```
PGREP(1)                                                     Use
r Commands                                                     P
GREP(1)

NAME
       pgrep, pkill - look up or signal processes based on
 name and other attributes

SYNOPSIS
       pgrep [options] pattern
       pkill [options] pattern

DESCRIPTION
       pgrep  looks through the currently running processe
s and lists the process IDs which match the selection crit
eria to
       stdout.  All the criteria have to match.  For examp
le,

              $ pgrep -u root sshd

       will only list the processes called sshd AND owned
by root.  On the other hand,
ual page pgrep(1) line 1 (press h for help or q to quit)
```

Discussion : A Program does nothing unless its instructions are executed by a CPU. A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.