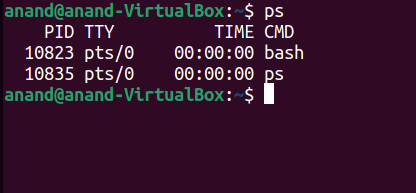
**Q.1 -** Use the ps, ps lx, ps tree and ps -aux command to display the process attributes.

**Answer:** We know Linux is a multitasking and multiprocessing system. We can view all information of a process, Linux provides a utility called ps(process state).

**:$ ps**



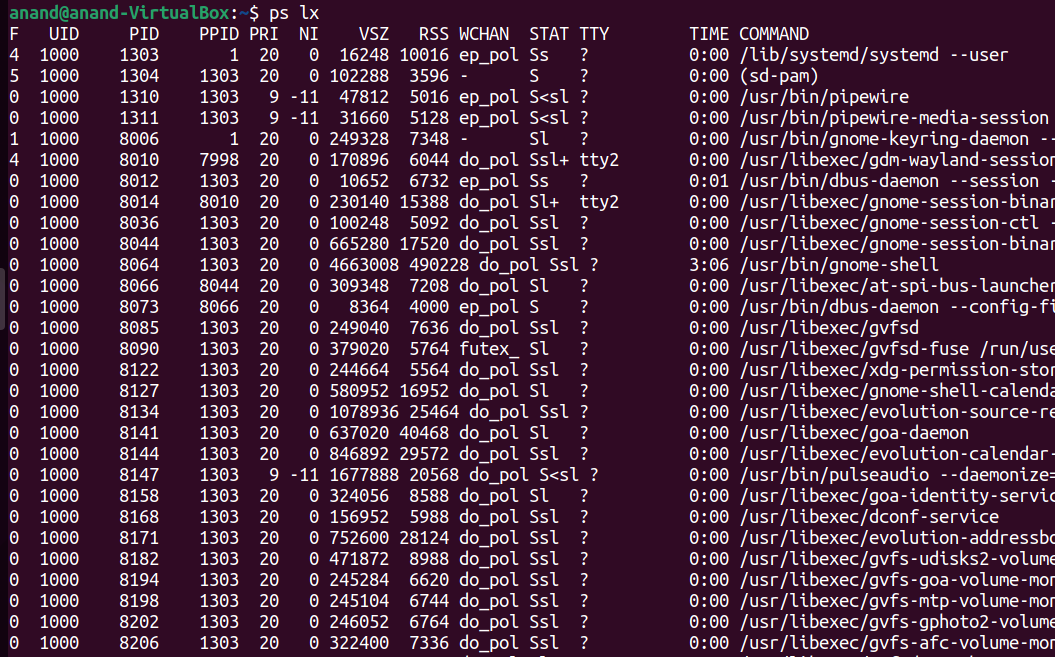
**PID:** Unique process id

**TTY:** TTY is a type of terminal where the current log is running.

**TIME:** Time used by the CPU to run the process.

**CMD:** Command name.

**:$ ps lx**



**UID:** User id.

**PID:** Uniqueue process id.

**PRI:** priority of the process.

**NI:** nice value, if the nice value is high then the priority of the process is low otherwise priority

High.

**VSZ:** virtual memory consumed by the process.

**RSS:** the size of physical memory consumed by the process.

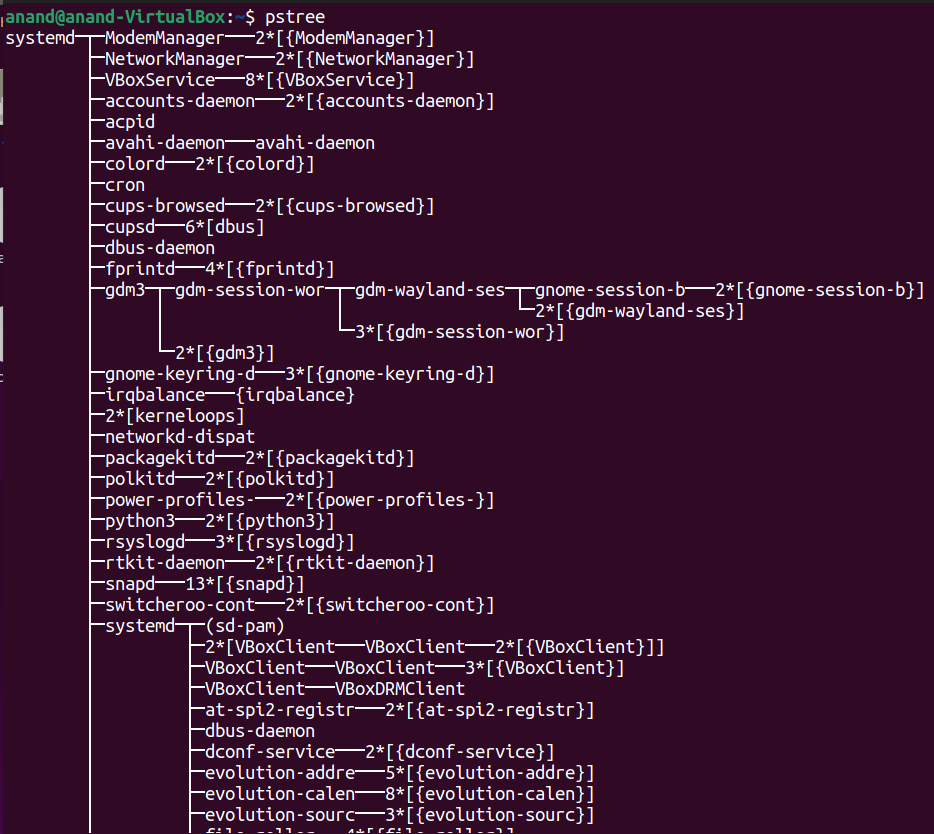
**STAT:** show the process state(running, idle, sleeping, stopped, zombie).

**TTY:** TTY is a type of terminal where the current log is running.

**TIME:** Time used by the CPU to run the process.

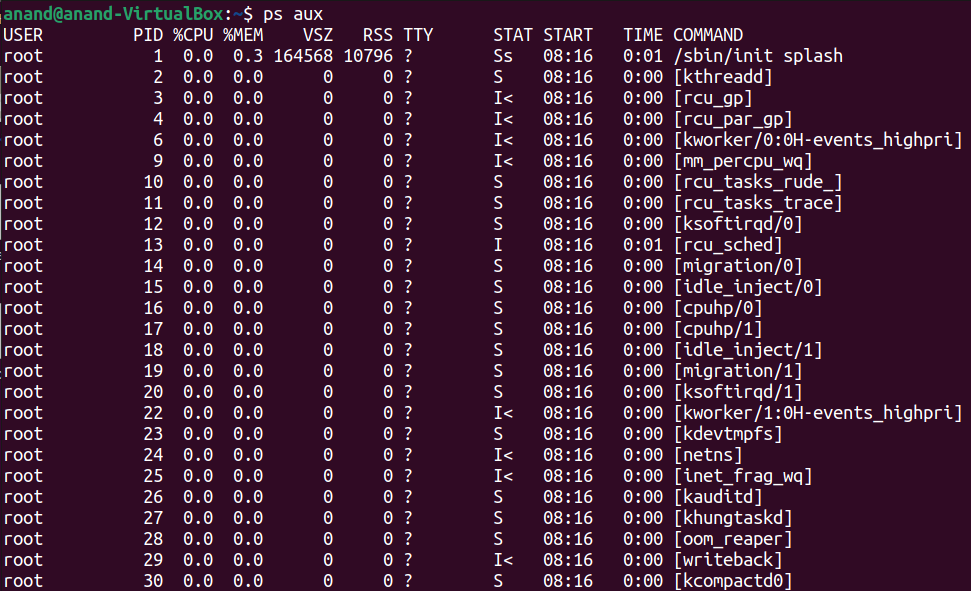
**CMD:** Command name.

**:$ pstree**



pstree displays the status of ongoing processes as a tree. If pid is omitted, the tree is rooted at either pid or init.

**:$ ps aux**



The ps aux command allows you to keep track of the processes that are running on your Linux system. Any programme executing on your system is associated with a process, which is used to manage and monitor the memory usage, processing time, and I/O resources of that programme.

**USER:** who run the process.

**PID:** Unique process id.

**%CPU:** show percentage of CPU utilization for a particular process.

**%MEM:** show percentage of Memory utilization for a particular process.

**VSZ:** virtual memory consumed by the process.

**RSS:** the size of physical memory consumed by the process.

**TTY:** TTY is a type of terminal where the current log is running.

**STAT:** show the process state(running, idle, sleeping, stopped, zombie).

**START**: show the time of the start running process.

**TIME:** Time used by the CPU to run the process.

**CMD:** Command name.

**Q.2 -** Learn the top command to display the resource utilization statistics of processes:

• Open a terminal and type the top command

• Start a browser and see the effect on the top display

• Compile a C program and observe the same effect (Use a long loop - say while(1) to

observe the effect)

• From the top display, answer the following: – How much memory is free in the system?

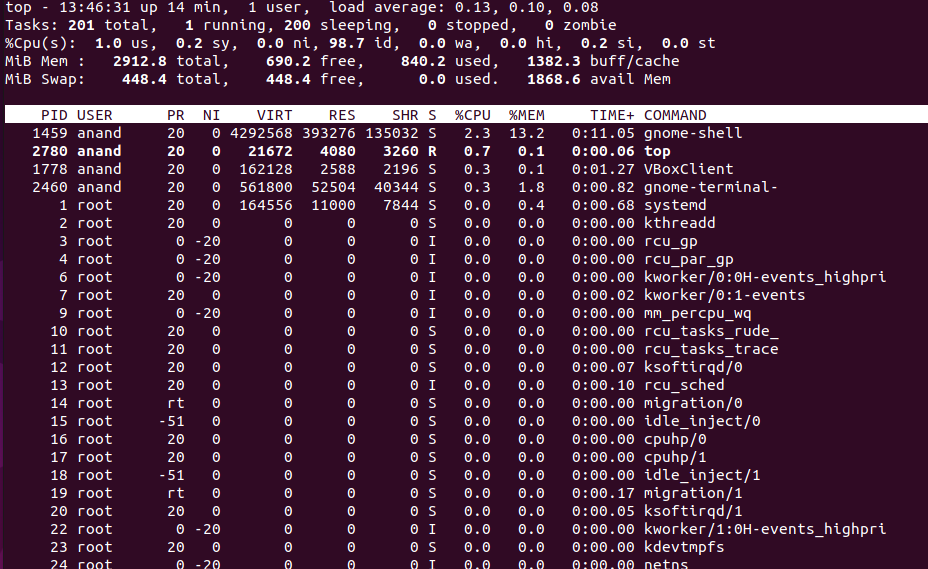
Which process is taking more CPU? – Which process has got maximum memory share?

• Write a CPU bound C program and an I/O bound C program (e.g. using more printf

statements within while(1) loop), compile and execute both of them.

**Answer:** This is used to display the Linux process, which gives a real-time view of the running system.

**:$ top**



It Divided two section

**Upper half-section display:**

**top:** show current time, uptime, active user, load average system perform work in 1 minute, 5 minutes and 15 minutes.

**Tasks:** Number of the total process, running process, sleeping process, stopped process and zombie process

**%Cpu(s):** CPU spent time in user space and kernel space, nice value shows the priority of the process, idle time, waiting time, hardware interrupt, software interrupt, steal time.

**MiB Mem:** showtotal RAM space, free space, used space, available buffer.

**MiB Swap:** show total swap memory, free space, used space, available space.

**Lower half-section display:**

**PID:** Unique process id.

**USER:** who run the process.

**PR:** priority of the process.

**NI:** nice value, if the nice value is high then the priority of the process is low otherwise priority

High.

**VIRT:** virtual memory consumed by the process.

**RES:** the size of physical memory consumed by the process.

**SHIR:** showShared Memory.

**S:** show the process state(running, idle, sleeping, stopped, zombie).

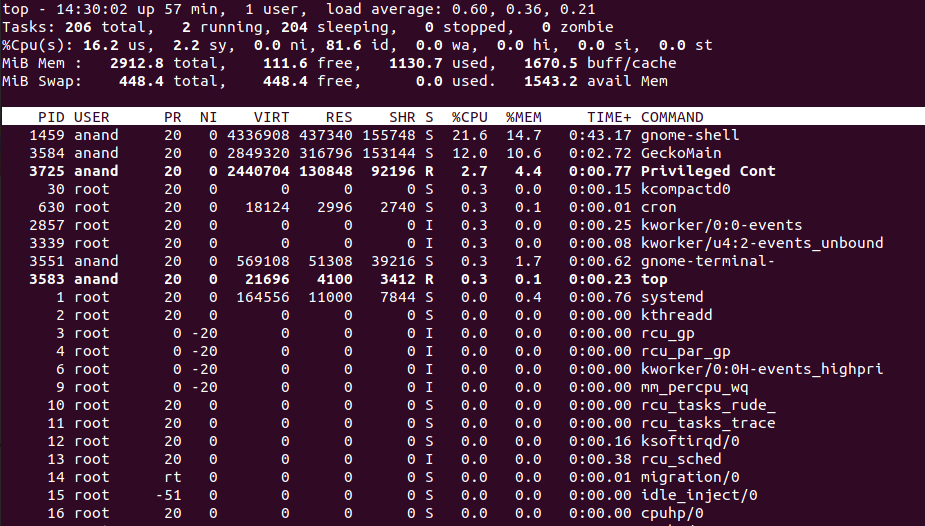
**%CPU:** show percentage of CPU utilization for a particular process.

**%MEM:** show percentage of Memory utilization for a particular process.

**TIME:** Time used by the CPU to run the process.

**CMD:** Command name.

**I Start a browser and see the effect on the top display:**



* %cpu more increase in userspace
* Increase uses memory by The RAM
* RAM free space decrease
* Buff/cache size increase
* In swap memory available space decrease

**Here I run a CPU bound program:**

#include<stdio.h>

int main()

{

int a=1;

while(1)

{

for(int i=0; i<100000; i++)

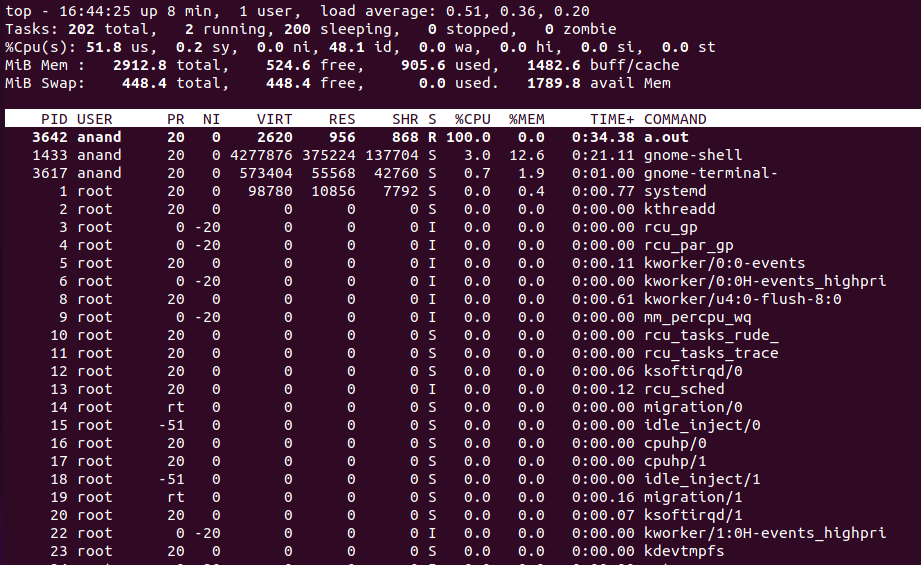
a++;

}

return 0;

}

**Here I can see changes using the top command:**



* COMMAND a.out take more CPU
* %cpu more increase in userspace
* Using the top command we can see 295.4 MB free space in physical memory and 448.2 MB free space in swap space.
* We can see above snapshot PID: 6170 and COMMAND: cpuBound take more CPU.
* We can see above snapshot PID: 1459 and COMMAND: gnome-shell has shared more memory.

**Here I run a I/O bound program:**

#include<stdio.h>

int main(

{

int a=1;

while(1)

{

for(int i=0; i<100000; i++)

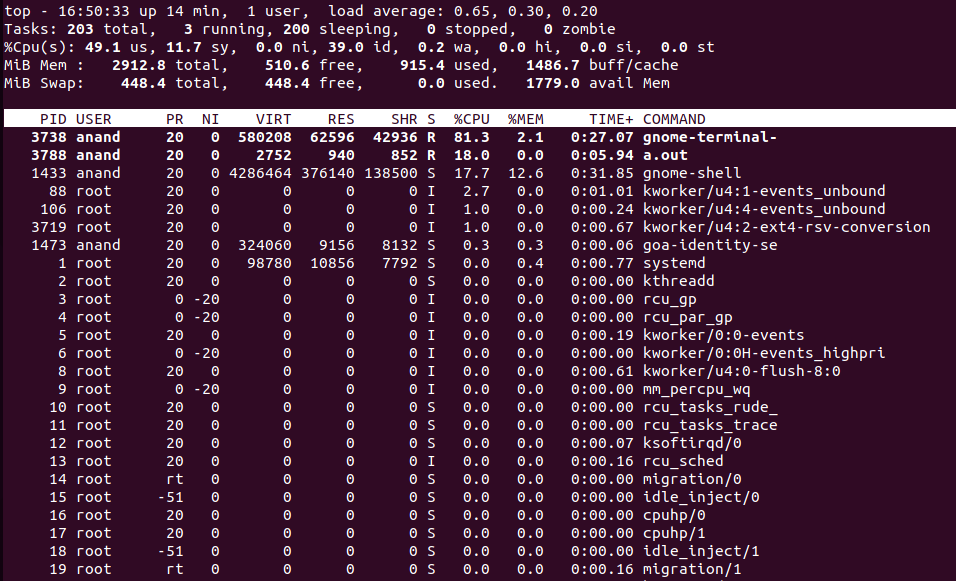
printf(" Anand ");

}

return 0;

}

**Here I can see changes using the top command:**



* COMMAND a.out does not take more CPU.
* %cpu more increase in userspace and also increase in kernel space.

**Q.3 -** Write a program in C that creates a child process, waits for the termination of the child and lists its PID, together with the state in which the process was terminated (in decimal and hexadecimal).

**Answer:** fork() is a system call, it creates a new process, a new process called the child process

Parent process completed then after chid process still running, its child process called zombie process. Waiting for child process execution completion wait() method required. zombie process not created Using wait() function.

Program: wait.c

#include<stdio.h>

#include<unistd.h>

int main()

{

int p=fork();

if(p<0)

printf("No created process");

else

if(p==0)

printf("Child Process ID : %d It's Parent process ID : %d\n",getpid(),getppid());

else

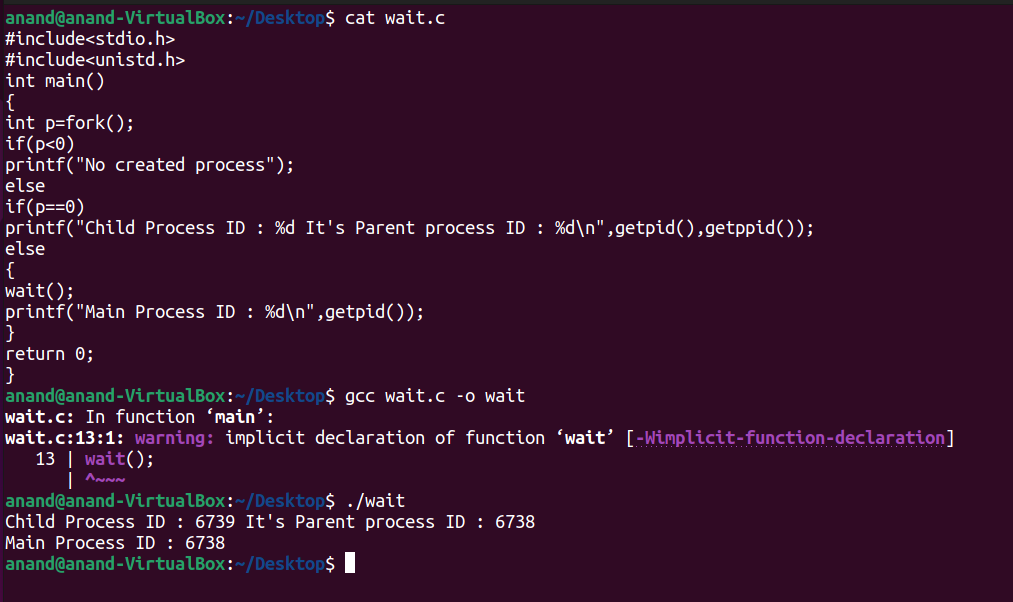
{

wait();printf("Main Process ID : %d\n",getpid());

}

return 0;

}



**Q.4 -** Write a C program such that it forks a new process. Then the parent process and the child process should create one more process such that the program in all has four running processes. Each process should print its process ID and its parent process ID. Draw process hierarchy starting from parent process.

**Answer:**

**Program**

#include<stdio.h>

#include<unistd.h>

int main()

{

int p1=fork();

int p2=fork();

if(p1<0 || p2<0)

printf("No created process");

else

if(p1==0)

{

if(p2==0)

printf("Child Process p2 PID : %d It's Parent process p2 ID : %d\n",getpid(),getppid());

else

{

wait();

printf("Parent Process P2 PID : %d\n",getpid());

}

}

else

{

{

wait();

if(p2==0)

printf("Child Process p1 PID : %d It's Parent process p1 ID : %d\n",getpid(),getppid());

else

{

wait();

printf("Parent Process P1 PID : %d\n",getpid());

}

}

}

return 0;

}

