

**NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR**

**Cachar, Assam**

**B.Tech. V<sup>th</sup> Sem**

**Subject Code:** CS-313

**Subject Name:** Operating System Lab

**Submitted By:**

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

- Q.1.**
- a. Write the functions of {ps, ps lx, ps -aux} commands in Linux.
  - b. What is the command to display all the processes under root?
  - c. What is the command to display the name of a process?
  - d. How to display the running processes in the terminal?
  - e. How to get the details of a particular process in the terminal?

**ANSWER:**

- a. Functions of {ps, ps lx, ps -aux} commands in Linux:

**I. ps :** This command-line utility is used to list the currently running processes and their corresponding information like PIDs. PS stand for Process Status.

**Example:** \$ ps -A

**II. ps lx :** This command-line utility is same as using ps. Here, the lx is a default value for ps. Using ps only gives extremely limited result, but adding lx as the format specifier lists all the processes and information.

**Example:** \$ ps lx

**III. ps -aux :** This command-line utility prints all the processes for all users. Here, 'x' means that the process is to be printed owned by a user named 'x'.

**Example:** \$ ps -auroot

- b. The command to display all the processes under root is \$ ps -u root
- c. The format specifier -p PID can be use along with ps to display the name of a process.

**Example:** \$ ps -p 3630

- d. The command to display the running processes in the terminal is \$ ps aux
- e. To get the details of a particular process in the terminal, we can use grep along with ps aux.

**Example:** \$ ps aux | grep -i firefox

**Q.2. a.** Learn and document the **top** command to display the resource utilisation statistics of processes. Try other variants of top command such as htop, atop, vtop. To install htop: *sudo apt-get install htop*

**b.** Compile the program cpu.c given to you and execute it. This program runs in an infinite loop without terminating. Now open another terminal, run the top command and answer the following questions about the cpu process.

- I.** What is the PID of the process running the cpu command?
- II.** How much CPU and memory does this process consume?
- III.** What is the current state of the process? For example, is it running or in a blocked state?

**ANSWER:**

**a.**

The top command-line utility is used to display the active Linux processes. It presents the list of top users of the user's system's resources- CPU shares and memory.

**Example:** \$ top

Pressing 't' on the terminal after executing top command gives the graph results in ASCII. Again pressing 't', the graphs get changed to solid block characters. Pressing 't' for the third time hides CPU display and task summary.

To display the summary results for a particular user, -u username format specifier can be used along with top.

**Example:** \$ top -u root

In the same process terminal, various keys combinations can be used to sort the running processes. For example, to sort by Process ID, press M and T keys. To sort by Memory usage, press M and P keys. To sort by running time, press M and T keys. Pressing C key displays the absolute path of the running process. The K key will kill the process highlighted without closing the top window. To sort by CPU utilisation, press Shift+P key. The T key gets the list of idle or sleeping processes.

There are various other better variants of top command-line utility namely: htop, atop, vtop, nmon, bashtop, gtop, glances, hagemon, gotop, ptop and many more. The primary role of these variants are the same as top, but they differ in interactive medium. These variants are better in a way that they provide mouse support, scrolling and various other advance interaction features. For instance, htop supports mouse point-click and scrolling along with much better looking process display in the terminal, which is eye-catchy and easier to understand. The alternative vtop utility is used to monitor system resource while having the ability to manage them. It also offers mouse support, unlike top, and looks like GUI in a terminal.

**b.**

```
// cpu.c

#include <unistd.h>
#include <stdio.h>

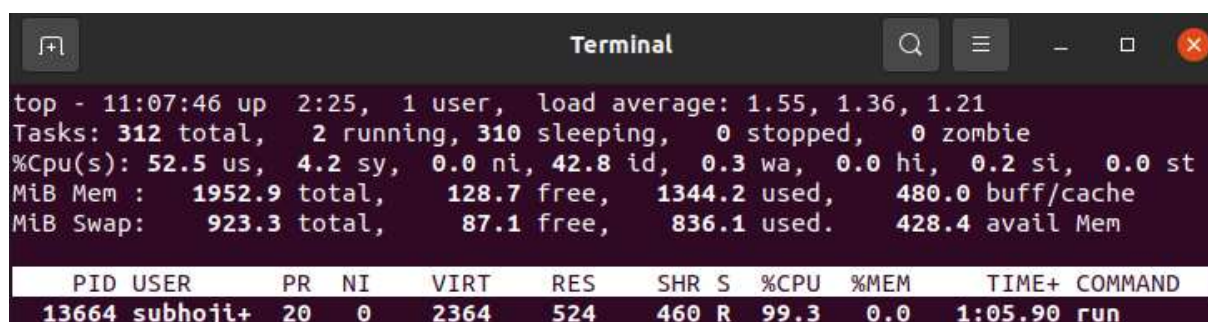
int main(int argc, char *argv[]) {
    unsigned int i,j;
    while(1) {
        j = 1;
        for(i = 1; i <= 10; i++)
            j = j*i;
    }
}

// gcc cpu.c -o run
// ./run
```

**I.** The PID for CMD:run is 10364

**II.** CMD:run is consuming 96%-100%, it's varying between 96.0% to 100.0%.  
However, it consume null memory, i.e., %MEM:0.0

**III.** The process has not been blocked. It is still in running state.



The image shows a terminal window titled "Terminal" with a search icon, a menu icon, and window control buttons. The terminal output displays the results of the 'top' command, showing system statistics and a list of running processes. The process list at the bottom shows PID 13664 for user 'subhoji' with 99.3% CPU usage and 0.0% memory usage.

```
top - 11:07:46 up 2:25, 1 user, load average: 1.55, 1.36, 1.21
Tasks: 312 total, 2 running, 310 sleeping, 0 stopped, 0 zombie
%Cpu(s): 52.5 us, 4.2 sy, 0.0 ni, 42.8 id, 0.3 wa, 0.0 hi, 0.2 si, 0.0 st
MiB Mem : 1952.9 total, 128.7 free, 1344.2 used, 480.0 buff/cache
MiB Swap: 923.3 total, 87.1 free, 836.1 used. 428.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
13664	subhoji	20	0	2364	524	460	R	99.3	0.0	1:05.90	run

**Q.3.** Document and present the **proc** file system of Linux, **iostat** command and **pmap** tool.

To use pmap, one needs to know the process ID of the process of interest. Thus run *ps auxw* to list the processes; then pick one of them. Now, run pmap using various flags like **pmap -X pid** to reveal details about the process. What you observe? How many different entities make up the address space?

**ANSWER:**

Proc file system (procfs) in Linux is a virtual file system created on fly when system boots and is dissolved at time of system shut down. The proc file system provides a communication medium between the kernel space and the user space. Also regarded as the control and information centre for kernel, the proc file system contains the useful information about the processes that are currently running.

The command for the same is: `$ ls -l /proc`

Using `$ ps -aux`, we can obtain PID of any running process, which can be further used in /proc file system such: `$ ls -ltr /proc/PID`

**For Example:** `$ ls -ltr /proc/11224`

The iostat command-line utility is used to monitor system I/O statistics for devices and partitions. It is included in sysstat package and hence first needs to be installed using *apt-get install sysstat*. The iostat command monitors the system I/O by observing the time the devices are active in relation to their average transfer rates.

**Example:** `$ iostat -xc`

The pmap command-line utility is used to display the memory map of a process. Hence, to use this command, the process ID of the process of interest must be known. To same can be obtained using ps command line utility such `$ ps -aux`. After finding the PID of the process, pmap command can be used such: `$ pmap -X PID`. This will display the memory map of the process of the corresponding PID. A memory map indicates how memory is spread out.

**Example:** `$ pmap -X 1099`

This gives the output path address of the process with PID:1099. The address obtained here is `/usr/lib/upower/upowerd`. There is no definite entities making up the address space, rather an address space defines a range of discrete addresses, each of which correspond to a network host, peripheral device, disk sector, a memory cell or other logical or physical entity.

The pmap command can be used with other format specifiers as well. The specifier option `-x` is used to display the memory map in an extended format. The option `-p` is used to display the full path of the files. The option `-d` is used to display the device format. The option `-q` is used to ignore the column names while displaying the report of the memory map, and so on.

**Q.4.** Consider the two programs `memory1.c` and `memory2.c` uploaded in the Moodle. Compile and run them one after the other. Both programs allocate a large array in memory. One of them accesses the array and the other does not. Both programs pause before exiting to let you inspect their memory usage. You can inspect the memory used by a process with the `ps` command. In particular, the output will tell you what the total size of the “virtual” memory of the process is, and how much of this is actually physically resident in memory. The virtual memory of the process is the memory the process thinks it has, while the OS only allocates a subset of this memory physically in RAM.

You are to write the observations of yours after exercising this.

**ANSWER:**

`memory1.c`

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#define ARRAY_SIZE 1000000

int main() {
    int array[ARRAY_SIZE];
    int i;
    printf("\n\nProgram : 'memory_1'\n");
    printf("_____ \n");
    printf("\n\nPID : %d \n",getpid());
    printf( "Size of int : %ld \n",sizeof(int));
    printf("\nPress Enter Key to exit.\n");
    getchar();
    return 0;
}
```

Program : ‘memory\_1’

PID: 15272

Size of int : 4

This program does not access the array.

The total virtual memory allocated to the process 15272 is 6284KB, which is only about 0.2% of the Memory usage, while the memory physically used in RAM is 4880KB.

memory2.c

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#define ARRAY_SIZE 1000000

int main() {
    int array[ARRAY_SIZE];
    int i;
    printf("\n\nProgram : 'memory_2'\n");
    printf("_____ \n");
    printf("\n\nPID : %d \n",getpid());
    printf( "Size of int : %ld \n",sizeof(int));
    for(i=0;i<ARRAY_SIZE/2;i++)
        array[i] = 10;
    for(i=1;i<ARRAY_SIZE/2;i++)
        array[i] = array[i-1]+25;
    printf("\nPress Enter Key to exit.\n");
    getchar();
    return 0;
}
```

Program : 'memory\_2'

PID : 15899

Size of int : 4

This program accesses the array.

The total virtual memory allocated to the process 15899 is 6284KB, same as for the program that does not access the array, which is only about 0.2% of the Memory usage, while the memory physically used in RAM is 4972KB, which is larger than what the previous program had physically used in the RAM.

From these two programs, we can conclude that the virtual memory allocated by the system is always larger than the physical memory actually needed or used.

**Q.5.** Demonstrate the parent and child processes through **top tool**. The parent and child programs should be written by you.

**ANSWER:**

```
#include <stdio.h>

int main () {
    for (int i=0; i<5; ++i) {
        if (fork () == 0) {
            printf ("Child PID => %d\tParent PID => %d\n", getpid(),
getppid());

            exit (0);
        }
    }

    for (int i=0; i<5; ++i)
        wait (NULL);

    getchar ();
}
```

Child PID => 17602                  Parent PID => 17601

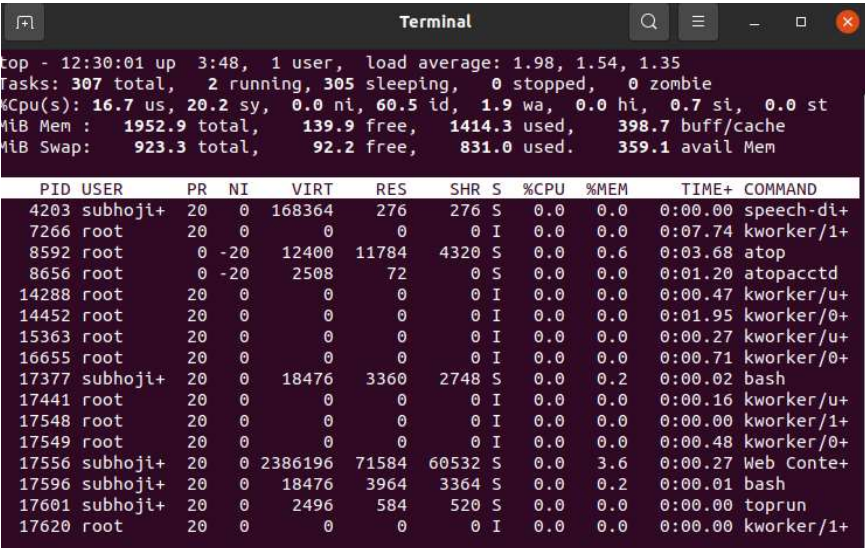
Child PID => 17604                  Parent PID => 17601

Child PID => 17605                  Parent PID => 17601

Child PID => 17603                  Parent PID => 17601

Child PID => 17606                  Parent PID => 17601

COMMND : toprun



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4203	subhoji+	20	0	168364	276	276	S	0.0	0.0	0:00.00	speech-di+
7266	root	20	0	0	0	0	I	0.0	0.0	0:07.74	kworker/1+
8592	root	0	-20	12400	11784	4320	S	0.0	0.6	0:03.68	atop
8656	root	0	-20	2508	72	0	S	0.0	0.0	0:01.20	atopacctd
14288	root	20	0	0	0	0	I	0.0	0.0	0:00.47	kworker/u+
14452	root	20	0	0	0	0	I	0.0	0.0	0:01.95	kworker/0+
15363	root	20	0	0	0	0	I	0.0	0.0	0:00.27	kworker/u+
16655	root	20	0	0	0	0	I	0.0	0.0	0:00.71	kworker/0+
17377	subhoji+	20	0	18476	3360	2748	S	0.0	0.2	0:00.02	bash
17441	root	20	0	0	0	0	I	0.0	0.0	0:00.16	kworker/u+
17548	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/1+
17549	root	20	0	0	0	0	I	0.0	0.0	0:00.48	kworker/0+
17556	subhoji+	20	0	2386196	71584	60532	S	0.0	3.6	0:00.27	Web Conte+
17596	subhoji+	20	0	18476	3964	3364	S	0.0	0.2	0:00.01	bash
17601	subhoji+	20	0	2496	584	520	S	0.0	0.0	0:00.00	toprun
17620	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/1+