Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 11

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 23-09-2020

Date of Submission:

Reference: Tazneen Akter (The mother board of my laptop is suddenly damaged and for that, I wasn't able to do my lab (2,3,5,6). So I took her help).

Submitted by

Name: Jannatul Ferdush Dhina

ID:IT-18012

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment no : 03

Experiment Name : Threads on Operating System.

Theory : Thread is a single sequence stream within a process. Threads have same properties as of the process so they are called as light weight processes. Threads are executed one after another but gives the illusion as if they are executing in parallel. Each thread has different states. Each thread has

1. A program counter

2. A register set

3. A stack space

Types of Threads:

User Level thread (ULT) –

Is implemented in the user level library, they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.

Kernel Level Thread (KLT) –

Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.

Working Process:

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
```

```
pthread_t tid[2];
void* doSomeThing(void *arg)
{
 unsigned long i = 0;
 pthread_t id = pthread_self();
 if(pthread_equal(id,tid[0]))
 {
 printf("\n First thread processing\n");
 }
 else
 {
 printf("\n Second thread processing\n");
 }
 for(i=0; i<(0xFFFFFFFF);i++);
 return NULL;
}
int main(void)
{
 int i = 0;
 int err;
 while(i < 2)
 {
 err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
 if (err != 0)
 printf("\ncan't create thread :[%s]", strerror(err));
 else
 printf("\n Thread created successfully\n");
 i++;
 }
 sleep(5);
 return 0;
}
```

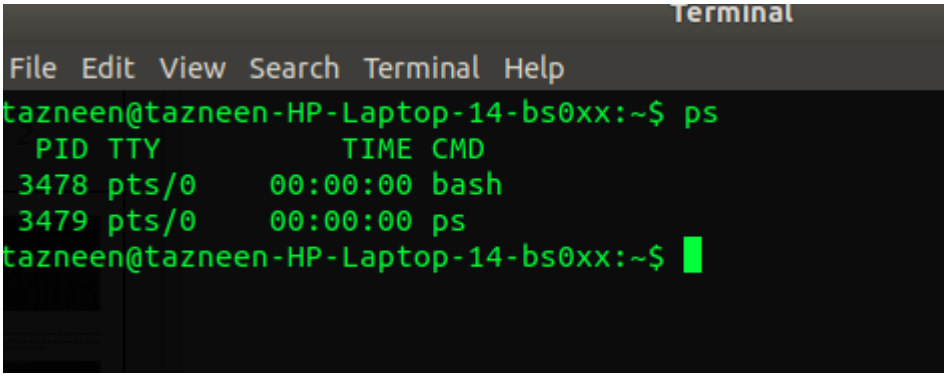
Output:

```
Terminal
File  Edit  View  Search  Terminal  Help
tazneen@tazneen-HP-Laptop-14-bs0xx:~$ cd Tazneen
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$ gcc thread.c -lpthread
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$ ./a.out
 Thread created successfully
 First thread processing
 Thread created successfully
 Second thread processing
tazneen@tazneen-HP-Laptop-14-bs0xx:~/Tazneen$
```

Thread in command line:

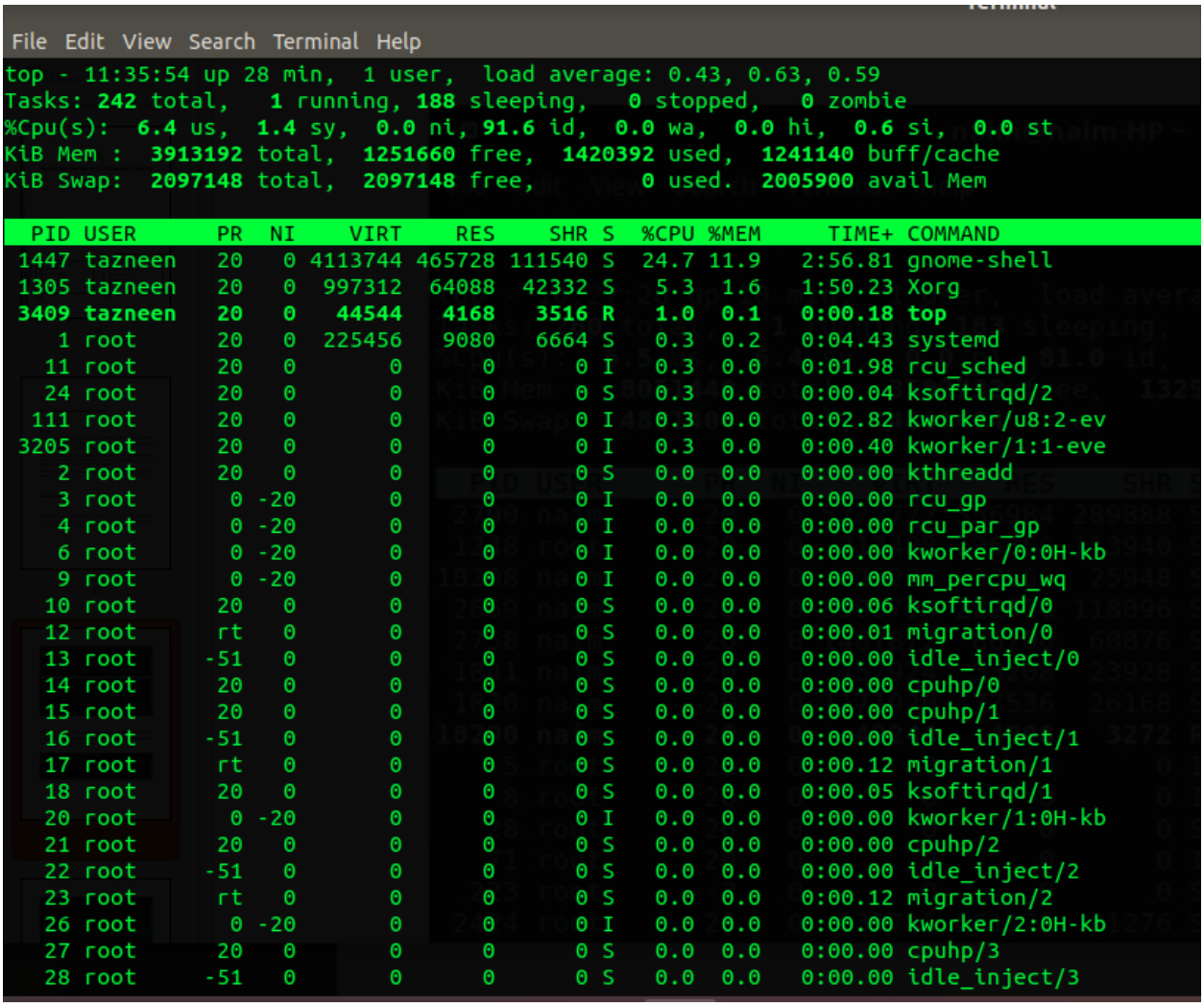
1.ps-

In ps command, "-T" option enables thread views. The following command list all threads created by a process with <pid>



2.top-

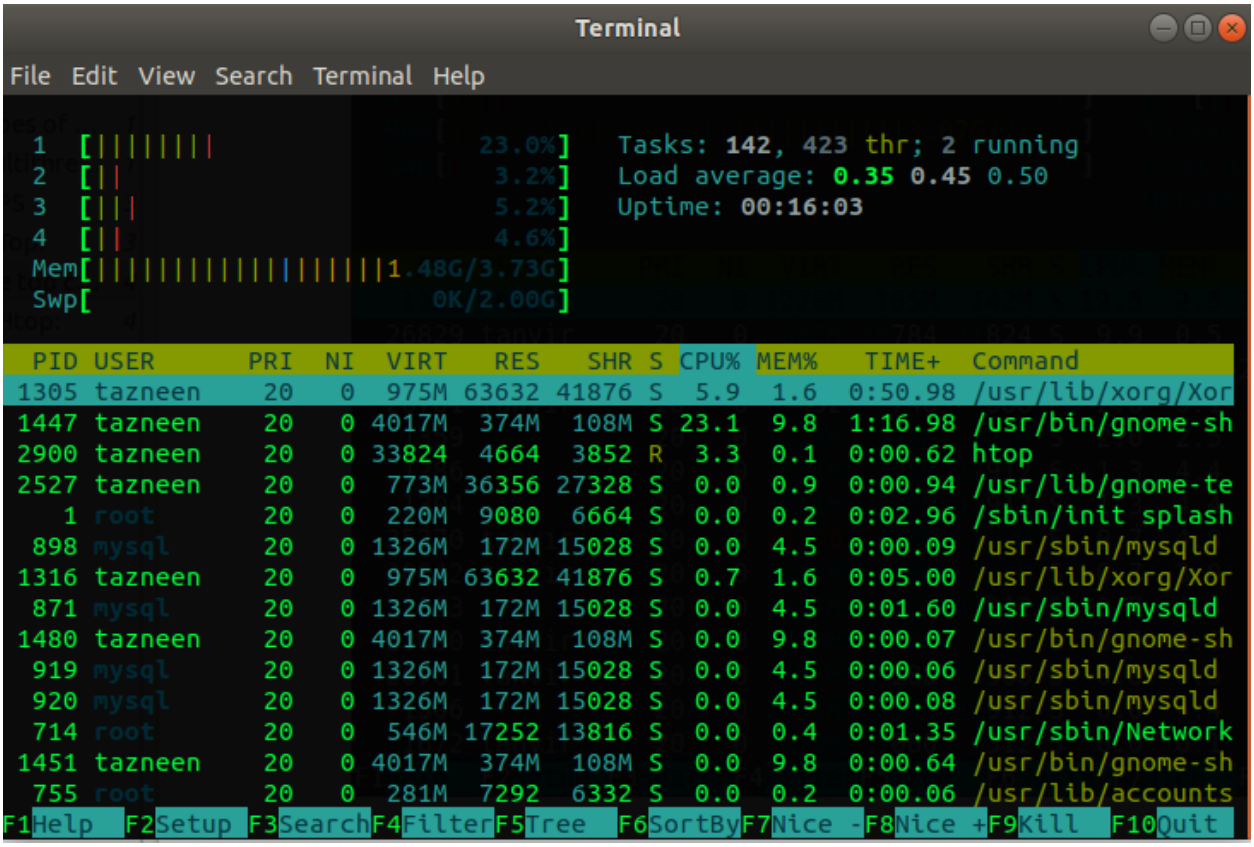
The top command can show a real-time view of individual threads. To enable thread views in the top output, invoke top with "-H" option. This will list all Linux threads.



To restrict the top output to a particular process and check all threads running inside the process: then we use $ top -H -p

3.htop-

A more user-friendly way to view threads per process is via htop, an ncurses-based interactive process viewer. This program allows you to monitor individual threads in tree views.



Discussion: In this lab we have implemented Threads on operating system using c language. By solving this problem we learn about thread in command line .