



MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

LAB REPORT

Department of : Information & Communication Technology

Lab Report No : 05

Lab Report On : **Connecting a database (MySQL) with linux**

Course Title : Operating Systems Lab

Course Code : ICT - 3110

Submitted by,

Name : Zahid Hasan Chowdhury

Student ID : IT-18017

Session : 2017-18

Year: 3rd

Semester : 1st

Dept. of ICT
MBSTU

Submitted to,

Nazrul Islam

Assistant Professor

Dept. of ICT,
MBSTU

Objectives:

- i. What is Thread.
- ii. Types of Thread
- iii. Implementation of Thread

Theory : A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

What are the differences between process and thread?

Threads are not independent of one other like processes as a result threads shares with other threads their code section, data section and OS resources like open files and signals. But, like process, a thread has its own program counter (PC), a register set, and a stack space.

Types of Thread:

Threads are implemented in following two ways –

- **User Level Threads** – User managed threads.
- **Kernel Level Threads** – Operating System managed threads acting on kernel, an operating system core.

Multithreading Models:

Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Corresponding Code:

```
#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:

```
zahid@zahid: ~/Documents
File Edit View Search Terminal Help
zahid@zahid: ~$ cd Documents/
zahid@zahid: ~/Documents$ com
combinediff          compare-im6           compose
comm                 compare-im6.q16        composite
command              compgen               composite-im6
command_not_found_handle complete            composite-im6.q16
compare              compopt
zahid@zahid: ~/Documents$ gcc commands.c -lpthread
zahid@zahid: ~/Documents$ ./a.out

Thread created successfully
First thread processing
Thread created successfully
Second thread processing
zahid@zahid: ~/Documents$ _
```

Thread in command line:

Here are several ways to show threads for a process on Linux.

1: PS

```
zahid@zahid: ~/Documents
File Edit View Search Terminal Help
zahid@zahid:~/Documents$ ps
  PID TTY      TIME CMD
 3416 pts/0    00:00:00 bash
 3636 pts/0    00:00:00 ps
zahid@zahid:~/Documents$ ps -T -p 3416
  PID  SPID TTY      TIME CMD
 3416  3416 pts/0    00:00:00 bash
zahid@zahid:~/Documents$ _
```

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

The "SID" column represents thread IDs, and "CMD" column shows thread names.

2: Top:

```

zahid@zahid: ~/Documents
File Edit View Search Terminal Help
top - 17:37:30 up 19 min, 1 user, load average: 0.83, 0.72, 0.71
Tasks: 317 total, 1 running, 240 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.4 us, 0.8 sy, 0.0 ni, 97.5 id, 0.1 wa, 0.0 hi, 0.2 si, 0.0 st
KiB Mem : 8032424 total, 3074252 free, 2026784 used, 2931388 buff/cache
KiB Swap: 4883452 total, 4883452 free, 0 used. 4403676 avail Mem

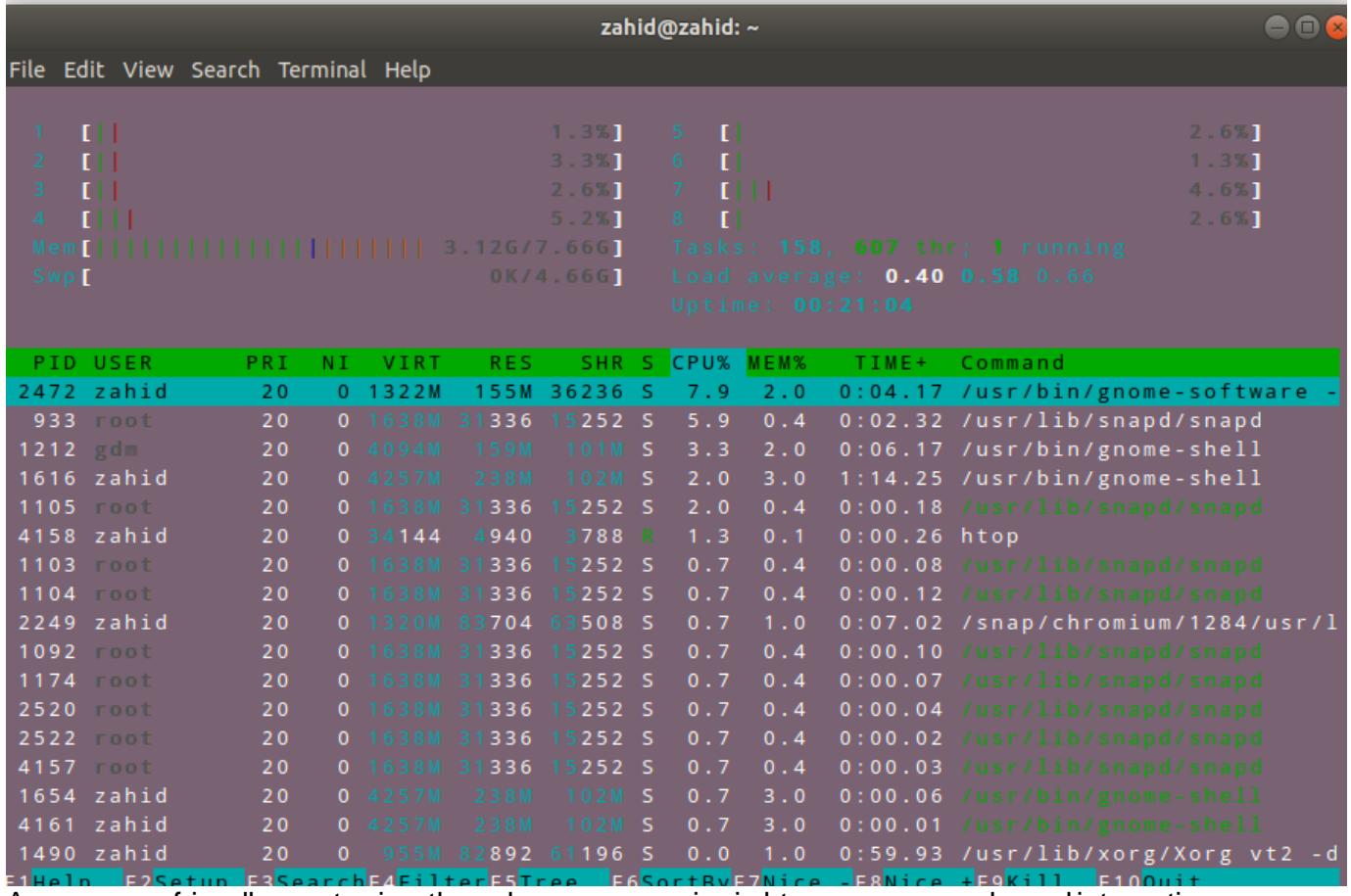
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1616 zahid 20 0 4360440 244040 105472 S 5.3 3.0 1:07.22 gnome-shell
3263 zahid 20 0 13.770g 358088 121952 S 4.6 4.5 0:16.57 chrome
1490 zahid 20 0 978304 82524 60964 S 3.0 1.0 0:55.27 Xorg
2244 zahid 20 0 1591568 414584 335952 S 1.7 5.2 1:29.88 chrome
3363 zahid 20 0 1290960 53796 38192 S 1.3 0.7 0:04.37 nautilus
1212 gdm 20 0 4193036 163508 104236 S 0.7 2.0 0:05.84 gnome-shell
3719 zahid 20 0 44544 4132 3416 R 0.7 0.1 0:00.19 top
1 root 20 0 225700 9536 6808 S 0.3 0.1 0:05.69 systemd
453 root -51 0 0 0 0 S 0.3 0.0 0:10.23 irq/109-ELAN130
1127 mysql 20 0 1424344 177864 15192 S 0.3 2.2 0:01.83 mysqld
2091 zahid 20 0 3911748 599116 372348 S 0.3 7.5 1:16.09 chrome
3312 root 20 0 0 0 0 I 0.3 0.0 0:01.42 kworker/5:1-eve
3656 root 20 0 0 0 0 I 0.3 0.0 0:00.06 kworker/7:1-eve
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kb
7 root 20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/0:1-eve
8 root 20 0 0 0 0 I 0.0 0.0 0:01.25 kworker/u16:0-i

```

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 `<pid>` and check all threads running inside the process: then we use \$ `top -H -p <pid>`

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.