



Mawlana Bhashani Science and Technology University

Lab-Report

Report No:03

Lab Report Name: Threads on Operating System.

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 20-09-2020

Date of Submission:

Submitted by

Name: Tazneen Akter

ID: IT-18056

3rd year 1st semester

Session: 2017-18

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
#include<unistd.h>
Thread created successfully
void* doSomething(void *arg)
First thread processing
unsigned long i = 0;
Thread created successfully;
if(pthread_equal(id,tid[0]))
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>

```
Terminal
File Edit View Search Terminal Help
tazneen@tazneen-HP-Laptop-14-bs0xx:~$ ps
  PID TTY          TIME CMD
 3478 pts/0    00:00:00 bash
 3479 pts/0    00:00:00 ps
tazneen@tazneen-HP-Laptop-14-bs0xx:~$ █
```

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.

```
File Edit View Search Terminal Help
top - 11:35:54 up 28 min, 1 user, load average: 0.43, 0.63, 0.59
Tasks: 242 total, 1 running, 188 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.4 us, 1.4 sy, 0.0 ni, 91.6 id, 0.0 wa, 0.0 hi, 0.6 si, 0.0 st
KiB Mem : 3913192 total, 1251660 free, 1420392 used, 1241140 buff/cache
KiB Swap: 2097148 total, 2097148 free, 0 used, 2005900 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM     TIME+ COMMAND
 1447 tazneen   20   0 4113744 465728 111540 S   24.7  11.9   2:56.81 gnome-shell
 1305 tazneen   20   0 997312  64088  42332 S    5.3   1.6   1:50.23 Xorg
 3409 tazneen   20   0 44544    4168   3516 R    1.0   0.1   0:00.18 top
    1 root      20   0 225456   9080   6664 S    0.3   0.2   0:04.43 systemd
   11 root      20   0      0      0      0 I    0.3   0.0   0:01.98 rcu_sched
   24 root      20   0      0      0      0 S    0.3   0.0   0:00.04 ksoftirqd/2
  111 root      20   0      0      0      0 I    0.3   0.0   0:02.82 kworker/u8:2-ev
 3205 root      20   0      0      0      0 I    0.3   0.0   0:00.40 kworker/1:1-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
    9 root       0 -20     0      0      0 I    0.0   0.0   0:00.00 mm_percpu_wq
   10 root      20   0      0      0      0 S    0.0   0.0   0:00.06 ksoftirqd/0
   12 root      rt    0      0      0      0 S    0.0   0.0   0:00.01 migration/0
   13 root     -51   0      0      0      0 S    0.0   0.0   0:00.00 idle_inject/0
   14 root      20   0      0      0      0 S    0.0   0.0   0:00.00 cpuhp/0
   15 root      20   0      0      0      0 S    0.0   0.0   0:00.00 cpuhp/1
   16 root     -51   0      0      0      0 S    0.0   0.0   0:00.00 idle_inject/1
   17 root      rt    0      0      0      0 S    0.0   0.0   0:00.12 migration/1
   18 root      20   0      0      0      0 S    0.0   0.0   0:00.05 ksoftirqd/1
   20 root       0 -20     0      0      0 I    0.0   0.0   0:00.00 kworker/1:0H-kb
   21 root      20   0      0      0      0 S    0.0   0.0   0:00.00 cpuhp/2
   22 root     -51   0      0      0      0 S    0.0   0.0   0:00.00 idle_inject/2
   23 root      rt    0      0      0      0 S    0.0   0.0   0:00.12 migration/2
   26 root       0 -20     0      0      0 I    0.0   0.0   0:00.00 kworker/2:0H-kb
   27 root      20   0      0      0      0 S    0.0   0.0   0:00.00 cpuhp/3
   28 root     -51   0      0      0      0 S    0.0   0.0   0:00.00 idle_inject/3
```

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.

```

Terminal
File Edit View Search Terminal Help

1 [|||||||] 23.0% Tasks: 142, 423 thr; 2 running
2 [||] 3.2% Load average: 0.35 0.45 0.50
3 [||] 5.2% Uptime: 00:16:03
4 [||] 4.6%
Mem[|||||||||] 1.48G/3.73G
Swp[|] 0K/2.00G

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1305 tazneen 20 0 975M 63632 41876 S 5.9 1.6 0:50.98 /usr/lib/xorg/Xor
1447 tazneen 20 0 4017M 374M 108M S 23.1 9.8 1:16.98 /usr/bin/gnome-sh
2900 tazneen 20 0 33824 4664 3852 R 3.3 0.1 0:00.62 htop
2527 tazneen 20 0 773M 36356 27328 S 0.0 0.9 0:00.94 /usr/lib/gnome-te
1 root 20 0 220M 9080 6664 S 0.0 0.2 0:02.96 /sbin/init splash
898 mysql 20 0 1326M 172M 15028 S 0.0 4.5 0:00.09 /usr/sbin/mysqld
1316 tazneen 20 0 975M 63632 41876 S 0.7 1.6 0:05.00 /usr/lib/xorg/Xor
871 mysql 20 0 1326M 172M 15028 S 0.0 4.5 0:01.60 /usr/sbin/mysqld
1480 tazneen 20 0 4017M 374M 108M S 0.0 9.8 0:00.07 /usr/bin/gnome-sh
919 mysql 20 0 1326M 172M 15028 S 0.0 4.5 0:00.06 /usr/sbin/mysqld
920 mysql 20 0 1326M 172M 15028 S 0.0 4.5 0:00.08 /usr/sbin/mysqld
714 root 20 0 546M 17252 13816 S 0.0 0.4 0:01.35 /usr/sbin/Network
1451 tazneen 20 0 4017M 374M 108M S 0.0 9.8 0:00.64 /usr/bin/gnome-sh
755 root 20 0 281M 7292 6332 S 0.0 0.2 0:00.06 /usr/lib/accounts
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit

```

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 .