

MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

**LAB REPORT**

Lab Report No : 03

Lab Report name : How to view threads of a process on linux and thread program

Course Title :Operating System Lab

Course Code :ICT-3110

Date of Performance :

Date of Submission : 28/9/2020

Submitted by,

Name : Farhana Afrin Shikha

ID :IT-18038

Session : 2017-18

3rd Year 1stsemester

Dept. of ICT

Submitted to,

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

**LAB 03 : How to view threads of a process on linux and thread program .**

**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.

**Types of Thread:** Threads are implemented in following two ways −

i.User Level Threads − User managed threads.

ii. 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

i. Many to many relationship.

ii.Many to one relationship.

Iii.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++;

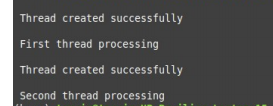
}

sleep(5);

return 0;

}

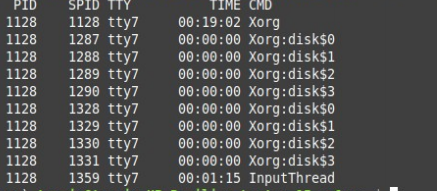
**Output:**



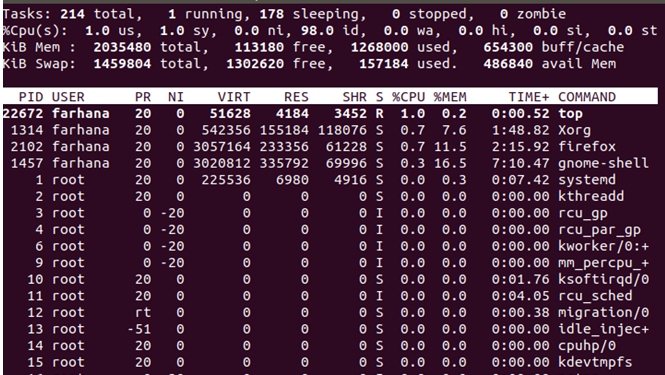
**Thread in command line:**

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

**1: PS**



**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.



Conclusion:I would like to monitor the number of threads used by a specific process on Linux. In the [Linux](https://en.wikipedia.org/wiki/Linux) operating system, **LinuxThreads** was a partial implementation of [POSIX Threads](https://en.wikipedia.org/wiki/POSIX_Threads) introduced in 1996. Threads are a popular modern programming abstraction. They provide multiple threads of execution within the same program in a shared memory address space. They can also share open files and other resourcesLinux has a unique implementation of threads. To the Linux kernel, there is no concept of a thread. Linux implements all threads as standard processes. The Linux kernel does not provide any special scheduling semantics or data structures to represent threads. Instead, a thread is merely a process that shares certain resources with other processes. Each thread has a unique task\_struct and appears to the kernel as a normal processLinuxThreads had a number of problems, mainly owing to the implementation, which used the [clone](https://en.wikipedia.org/wiki/Clone_(Linux_system_call)) system call to create a new [process](https://en.wikipedia.org/wiki/Process_(computing)) sharing the parent's [address space](https://en.wikipedia.org/wiki/Address_space). To improve the situation, two competing projects were started to develop a replacement; [NGPT](https://en.wikipedia.org/w/index.php?title=NGPT&action=edit&redlink=1) (Next Generation POSIX Threads) and [NPTL](https://en.wikipedia.org/wiki/NPTL).  [NPTL](https://en.wikipedia.org/wiki/NPTL) won out and is today shipped with the vast majority of Linux systems.