**Lab No : 03**

**Name of the lab report :** Threads on Operating System

**ID: IT-17018**

**Objective:** Learning aboutThreads, types of threads, how threads get implemented & worked on our Operating System.

**Threads on Operating System**

**1. What is Thread ?**

**Answer** **:**

Thread is an execution unit which consists of its own program counter, a stack, and a set of registers. Threads are also known as Lightweight processes. Threads are popular way to improve application through parallelism. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel. As each thread has its own independent resource for process execution, multiple processes can be executed parallel by increasing number of threads.

**2. Types of thread with explain ?**

**Answer :**

There are two types of threads:

1. User Threads
2. Kernel Threads

**User Level Threads:**

User threads, are above the kernel and without kernel support. These are the threads that application programmers use in their programs. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread.

**Advantages:**

1. User level threads does not require modification to operating systems.
2. User level threads are cheap and fast.
3. User level thread does not require kernel mode privileges. They can run on any operating system.

**Disadvantages:**

1. If one user level thread perform blocking operation then entire process will be blocked.
2. There is a lack of coordination between threads and operating system kernel.

**Kernel Level Threads:**

Kernel threads are supported within the kernel of the OS itself. All modern OSs support kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously. In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded.

**Advantages:**

1. If one kernel thread perform blocking operation then another thread can continue execution.

2. Scheduler may decide to give more times to a process having large number of threads than process having small number of threads.

3. Kernel threads can simultaneously schedule multiple threads from the same process or multiple process.

**Disadvantages:**

1. Kernel threads are slower to create and manage than user level.
2. It requires a full Thread Control Block(TCB) for each thread to maintain information about threads.

**3. Implementation of Threads :**

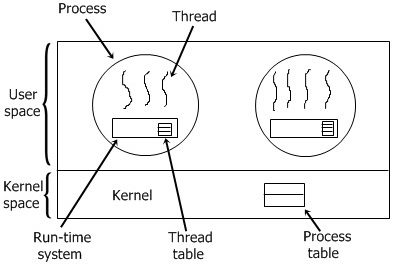
**Answer :**

There are the following two ways basic on the operating system.

1. Implementation in user space
2. Implementation in kernel

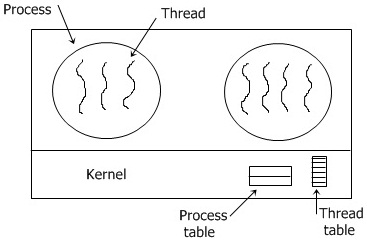
**Threads Implementation in User Space:**

A user-level threads package can be executed on an operating system that doesn't support threads and this is the main advantage of this implementation model i.e. In this model of implementation, the threads package entirely in user space, the kernel has no idea about it. Threads package in user space. The first, and most obvious, advantage is that a user-level threads package can be implemented on an operating system that does not support threads.



**Threads Implementation in Kernel:**

In this method of implementation model, the threads package completely in the kernel. There is no need for any runtime system. The thread table of the kernel holds each registers, state, and some other useful information of the thread . In this method of implementation model, the threads package completely in the kernel. To maintain the record of all threads in the system a kernel has a thread table. A call to the kernel is made whenever there is a need to create a new thread or destroy an existing thread. In this, the kernel thread table is updated.



**Conclusion:**

I learned about threads, thread types, and thread implementation by Through this lab class . From this lab class I also learn about Kernel-level threads much better, but still significant overhead and User-level threads even better, but not well integrated with OS. Now, we can get our threads to correctly cooperate with each other.