**Lab Report No: 03**

**Name of the Lab: Threads on Operating System.**

**1.What is Thread?**

A thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

When we launch a new app or run a new program, the OS creates a new process for it and creates a new thread called as the main thread. All the operations of the process are done in the main thread such as responding to events such as clicks unless the process requests for a new thread to be created. Thread can be thought of as a light weight process.

**2. Types of Threads**

There are two types of threads in operating system. They are:

1.User level thread 2. Kernel level thread

**User level thread:**

User level threads are managed by a user level library. However, they still require a kernel system call to operate. It doesn’t mean that the kernel knows anything about thread management. It only takes care of the execution part. Thread library contains code for creating, destroying, threads and passing message or data between threads. User level threads are typically fast.

**Advantages**

* Kernel-mode privilege does not require for thread switching.
* These threads are fast to create and manage.
* User level thread works even if the OS does not support threads.
* User level threads are more portable.
* Threading library controls the flow of thread.

**Disadvantages**

* If thread blocks, the Kernel may block all threads
* Not suitable for the multiprocessor system.
* User level threads also do not support system-wide scheduling priority.

**Kernel Level Thread:**

A Kernel thread, sometimes called a light weight process, is created and scheduled by the kernel. Supporting thread at kernel level means that operating system is multithreaded. Operating system kernel maintains thread abstractions, synchronization and scheduling. It allows the resource to share.

**Advantages**

* Each thread can be treated separately.
* A thread blocking in the Kernel does not block all other threads in the same process.
* Kernel routines itself as multithreaded.

**Disadvantages**

* Slower than the user level thread.
* There will be overhead and increased in Kernel complexity.

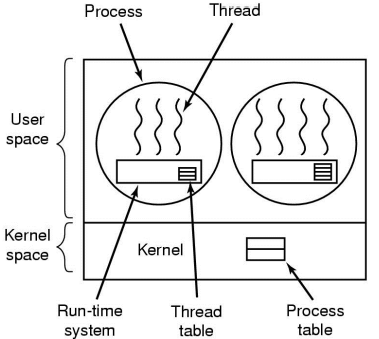
**3.Implementation of Threads:**

There are two ways of implementing a thread package:

* Threads implementation in user space
* Threads implementation in kernel

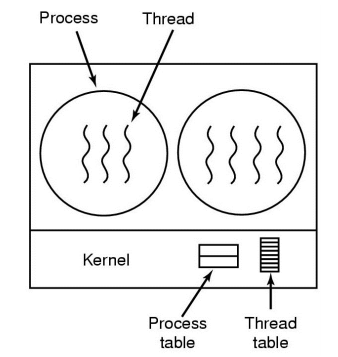
**Threads implementation in user space**

The first method is to put the threads package entirely in user space. The kernel knows nothing about them. As far as the kernel is concerned, it is managing ordinary, single-threaded processes. 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. All operating systems used to fall into this category, and even now some still do



**Threads implementation in kernel**

The kernel's thread table holds each thread's registers, state, and other information. The information is the same as with user-level threads, but it is now in the kernel instead of in user space (inside the run-time system). This information is a subset of the information that traditional kernels maintain about each of their single-threaded processes, that is, the process state. In addition, the kernel also maintains the traditional process table to keep track of processes.



**Conclusion:**

From this lab I have learnt mainly about Thread. Definition of Thread, Types of Thread and implementation process of Thread are the main topic of this lab. Threads provide a way to improve application performance through parallelism.