**Threads on Operating System**

**1. What is Thread ?**

**Ans :** Thread : A thread is a path of execution within a process . A process can contain multiple threads.

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.

**2. Types of Threads :**

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.

**User Level Threads :**

In this case, the thread management kernel is not aware of the existence of threads. 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 starts with a single thread.

**Advantages :**

1. Thread switching does not require Kernel mode privileges.
2. User level thread can run on any operating system.
3. Scheduling can be application specific in the user level thread.
4. User level threads are fast to create and manage.

**Disadvantages :**

1. In a typical operating system, most system calls are blocking.
2. Multithreaded application cannot take advantage of multiprocessing.

**Kernel Level Threads :**

In this case, thread management is 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. All of the threads within an application are supported within a single process. Kernel threads are generally slower to create and manage than the user threads.

**Advantages :**

1. Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
2. If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
3. Kernel routines themselves can be multithreaded.

**Disadvantages :**

1. Kernel threads are generally slower to create and manage than the user threads.
2. Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

**3. Implementation of Threads :**

There are two ways of implementing a thread package :

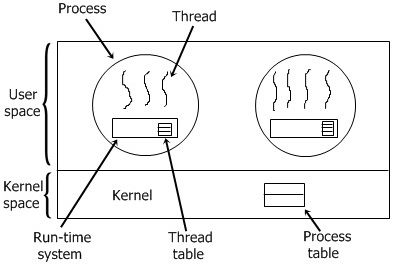
* Threads implementation in user space
* Threads implementation in kernel

Now describe briefly about the above two ways of implementing a thread package .

**Threads implementation in user space :**

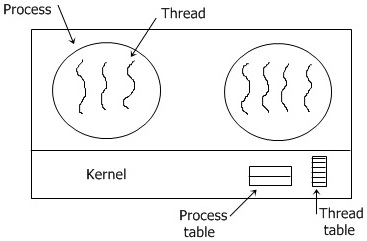
In this model of implementation, the threads package entirely in user space, the kernel has no idea about it. A user-level threads package can be executed on an operating system that doesn't support threads and this is the main advantage of this implementation model i.e. Threads package in user space.

All of these implementations have the same general structure as illustrated in the figure given below.



**Threads implementation in user space :**

In this method of implementing the threads package entirely in the kernel, no any run-time system is need in each as illustrated in the figure given below.



In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table .Whenever a thread wants to create a new thread or destroy an existing thread, then it makes a kernel call, which does the creation or destruction just by updating the kernel thread table .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. There is no need for any runtime system. 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.

**Other two methods are as follows:**

* Hybrid implementation
* Scheduler activation

**Hybrid implementation**

In this implementation, there is some set of user-level threads for each kernel level thread that takes turns by using it.

**Scheduler activation**

The objective of this scheduler activation work is to replicate the working or function of kernel threads, but with higher performance and better flexibility which are usually related to threads packages which are implemented in user space.

**Conclusion :**

In this lab we have learn about thread on operation system . we seen that every day there many task operate computer at a time . That is multithreading . That means many task execute by the computer . Thread is very important for modern computer technology.