Question no\_01>>> What are threads?

Answer to this question:

Threads are mechanism which run an application to perform many task in simultaneously.  It can be also defined as a set of processes that shares the same virtual memory as well as range of other attributes. Each thread is executing the same program code shares same text**,** data**,** heapsegments but have their own stacksegment**.**

Question no\_02>>> What are different threads?

Answer to this question:

There are two types of threads. One is user space and another is kernel space.

User space thread:

User-space avoids the kernel and manages the tables itself. Often this is called "cooperative multitasking" where the task defines a set of routines that get "switched to" by manipulating the stack pointer.

Kernel space thread:

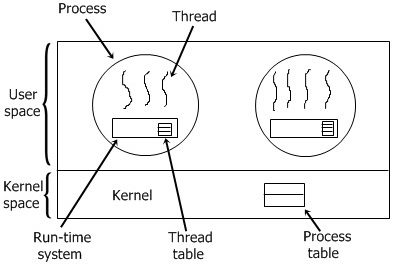
Kernel-space threads often are implemented in the kernel using several tables (each task gets a table of threads). In this case, the kernel schedules each thread within the timeslice of each process.

Question no\_03>>> What are Implementation of threads?

Answer to this question:

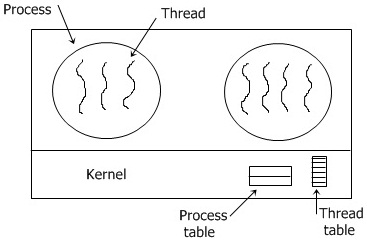
Implementation of user space thread:

In this model of implementing the threads package completely in user space, the kernel don't know anything about them. The advantage of implementing threads package in user space is that a user-level threads package can be implemented on an OS (Operating System) that doesn't support threads. All of these implementations have the same general structure as illustrated in the figure given below.



Implementation of kernel space thread:

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.



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.