S. No.	Parameters	**User Level Thread**	**Kernel Level Thread**
1.	**Implemented by**	User threads are implemented by users.	Kernel threads are implemented by Operating System (OS).
2.	**Recognize**	Operating System doesn't recognize user level threads.	Kernel threads are recognized by Operating System.
3.	**Implementation**	Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
4.	**Context switch time**	Context switch time is less.	Context switch time is more.
5.	**Hardware support**	Context switch requires no hardware support.	Hardware support is needed.
6.	**Blocking operation**	If one user level thread performs blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.
7.	**Multithreading**	Multithread applications cannot take advantage of multiprocessing.	Kernels can be multithreaded.
8.	**Creation and Management**	User level threads can be created and managed more quickly.	Kernel level threads take more time to create and manage.
9.	**Operating System**	Any operating system can support user-level threads.	Kernel level threads are operating system-specific.

S. No.	Parameters	User Level Thread	Kernel Level Thread
10.	**Thread Management**	The thread library contains the code for thread creation, message passing, thread scheduling, data transfer and thread destroying	The application code does not contain thread management code. It is merely an API to the kernel mode. The Windows operating system makes use of this feature.
11.	**Example**	Example: Java thread, POSIX threads.	Example: Window Solaris.
12.	**Advantages**	• User Level Threads are simple and quick to create. • Can run on any operating system • They perform better than kernel threads since they don't need to make system calls to create threads. • In user level threads, switching between threads does not need kernel mode privileges.	• Scheduling of multiple threads that belong to same process on different processors is possible in kernel level threads. • Multithreading can be there for kernel routines. • When a thread at the kernel level is halted, the kernel can schedule another thread for the same process.
13.	**Disadvantages**	• Multithreaded applications on user-level	• Transferring control within a process from

S. No.	Parameters	User Level Thread	Kernel Level Thread
		threads cannot benefit from multiprocessing. • If a single user-level thread performs a blocking operation, the entire process is halted.	one thread to another necessitates a mode switch to kernel mode. • Kernel level threads takes more time to create and manage than user level threads.