Contents

[Thread 2](#_Toc107312863)

[Advantage of Thread 2](#_Toc107312864)

[Types of Thread 2](#_Toc107312865)

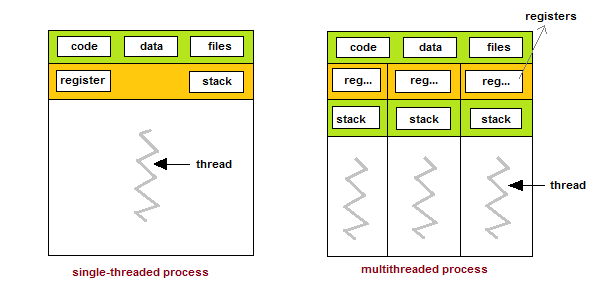
[Multithreading Models 3](#_Toc107312866)

[Three types of Thread 3](#_Toc107312867)

[Optimal Number of Threads 4](#_Toc107312868)

Thread

# Thread

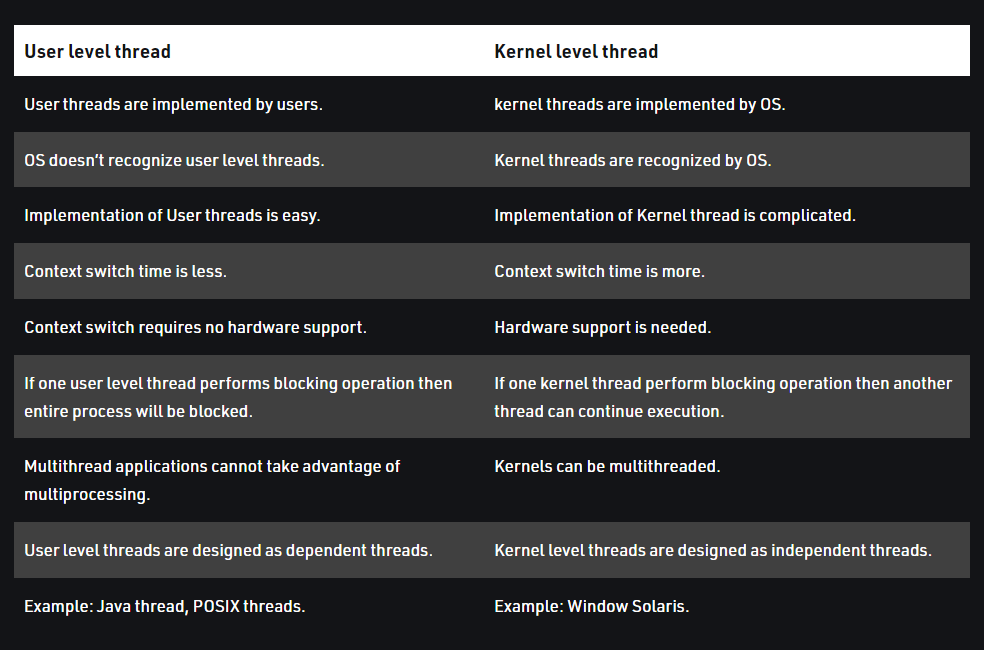


# Advantage of Thread

* Responsiveness
* Resource sharing, hence allowing better utilization of resources.
* Economy. Creating and managing threads becomes easier.
* Scalability. One thread runs on one CPU. In Multithreaded processes, threads can be distributed over a series of processors to scale.
* Context Switching is smooth. Context switching refers to the procedure followed by the CPU to change from one task to another.

# Types of Thread

* User Level Thread: The user-level threads are implemented by users and the kernel is not aware of the existence of these threads. It handles them as if they were single-threaded processes. User-level threads are small and much faster than kernel level threads. They are represented by a program counter(PC), stack, registers and a small process control block. Also, there is no kernel involvement in synchronization for user-level threads.
* Kernal Level Thread: Kernel-level threads are handled by the operating system directly and the thread management is done by the kernel. The context information for the process as well as the process threads is all managed by the kernel. Because of this, kernel-level threads are slower than user-level threads.



# Multithreading Models

The user threads must be mapped to kernel threads, by one of the following strategies:

* Many to One Model :
  + In the many to one model, many user-level threads are all mapped onto a single kernel thread.
* One to One Model
  + The one to one model creates a separate kernel thread to handle each and every user thread.
* Many to Many Model
  + The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.

# Three types of Thread

* POSIX Pitheads may be provided as either a user or kernel library, as an extension to the POSIX standard.
* Win32 threads are provided as a kernel-level library on Windows systems.
* Java threads: Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either Pitheads or Win32 threads depending on the system.

# Optimal Number of Threads

Not long ago, I was doing performance testing on a 2 quad-core machine running an ASP.NET application on Mono under a pretty decent load. We played with the minimum and maximum number of threads and in the end we found out that for that particular application in that particular configuration the best throughput was somewhere between 36 and 40 threads. Anything outside those boundaries performed worse. Lesson learned? If I were you, I would test with different number of threads until you find the right number for your application.

One thing for sure: 4k threads will take longer. That's a lot of context switches.