What Does Multithreading Mean?

Multithreading is a CPU (central processing unit) feature that allows two or more instruction threads to execute independently while sharing the same process resources. A thread is a self-contained sequence of instructions that can execute in parallel with other threads that are part of the same root process.

Multithreading allows multiple concurrent tasks can be performed within a single process. When data scientists are training machine learning algorithms, a multithreaded approach to programming can improve speed when compared to traditional parallel multiprocessing programs.

Even though it's faster for an operating system (OS) to switch between threads for an active CPU task than it is to switch between different processes, multithreading requires careful programming in order to avoid conflicts caused by race conditions and deadlocks.

To prevent race conditions and deadlocks, programmers use locks that prevent multiple threads from modifying the value of the same variable at the same time.

Techopedia Explains Multithreading

The 32- and 64-bit versions of Windows use pre-emptive multithreading in which the available processor time is shared. All threads get an equal time slice and are serviced in a queue-based model. During thread switching, the context of a pre-empted thread is stored and reloaded in the next thread in the queue. This takes so little time, that the running threads seem to execute in parallel.

How does Multithreading Work?

In programming, a thread maintains a list of information relevant to its execution, including the priority schedule, exception handlers, a set of CPU registers, and stack state in the address space of its hosting process. Threading can be useful in a single-processor system because it allows the primary execution thread to be responsive to user input while supporting threads execute long-running tasks in the background that do not require user intervention.

When thinking about how multithreading is done, it's important to separate the two concepts of parallel and concurrent processing.

Parallel multiprocessing means the system is actually handling more than one thread at a given time. Concurrent processing means that only one thread will be handled at a time, but the system will create efficiencies by moving quickly between two or more threads.

Another important thing to note is that for practical purposes, computer systems set up for human users can have parallel or concurrent systems, with the same end result – the process looks parallel to the user because the computer is working so quickly in terms of microseconds.

The evolution of multicore systems means that there is more parallelism, which alleviates the need for efficient concurrent processing. The development of faster and more powerful microchips and processors on this end of the expansion of Moore's law is important to this type of hardware design and engineering in general.

In addition, much of the parallel or concurrent processing is made available according to the vagaries of the operating system. So in effect, to the human user, either parallel or concurrent process, or processes that are mixed, are all experienced as parallelism in real-time.

Types of Multithreading

Different types of multithreading apply to various versions of operating systems and related controls that have evolved in computing: for example, in pre-emptive multithreading, the context switch is controlled by the operating system. Then there's cooperative multithreading, in which context switching is controlled by the thread. This could lead to problems, such as deadlocks if a thread is blocked waiting for a resource to become free.

Many other types of models for multithreading also apply, for example, coarse-grained, interleaved and simultaneous multithreading models will determine how the threads are coordinated and processed. Other options for multithreading include many to many, many to one and one to one models. Some models will use concepts like equal time slices to try to portion out execution among threads. The type of multithreading depends on the system itself, its philosophy and its build, and how the engineers planned multithreading functionality within it.

In the active/passive system model, one thread remains responsive to a user, and another thread works on longer-term tasks in the background. This model is useful for promoting a system that looks parallel from a user viewpoint, which brings us to a major point in evaluating processes like micro threading from both ends: from the perspective of the engineer, and the perspective of the end-user.

Multithreading vs. Multiprocessing

In programming, an instruction stream is called a thread and the instance of the computer program that is executing is called a *process*. Each process has its own memory space where it stores threads and other data the process requires to execute.

While multithreading allows a process to create more threads to improve responsiveness, multiprocessing simply adds more CPUs to increase speed.

'

BASIS FOR COMPARISON	MULTIPROCESSING	MULTITHREADING
Basic	Multiprocessing adds CPUs to increase computing power.	Multithreading creates multiple threads of a single process to increase computing power.
Execution	Multiple processes are executed concurrently.	Multiple threads of a single process are executed concurrently.