"User level Thread"	"Kernel Level Thread"
User level threads are managed by User level library	1) Kernel level threads are managed by OS System Calls
User level threads are typically fast	2) Kernel level threads are slower than User level
Context Switching is faster	3) Context Switching is slower
If one user level threads perform blocking operation then entire process get blocked	4) If one Kernel level thread blocked, No affect on others.

Multithreading Models and Hyperthreading

Types of Threads:

1) User Threads - Supported above the kernel and are managed without kernel support.

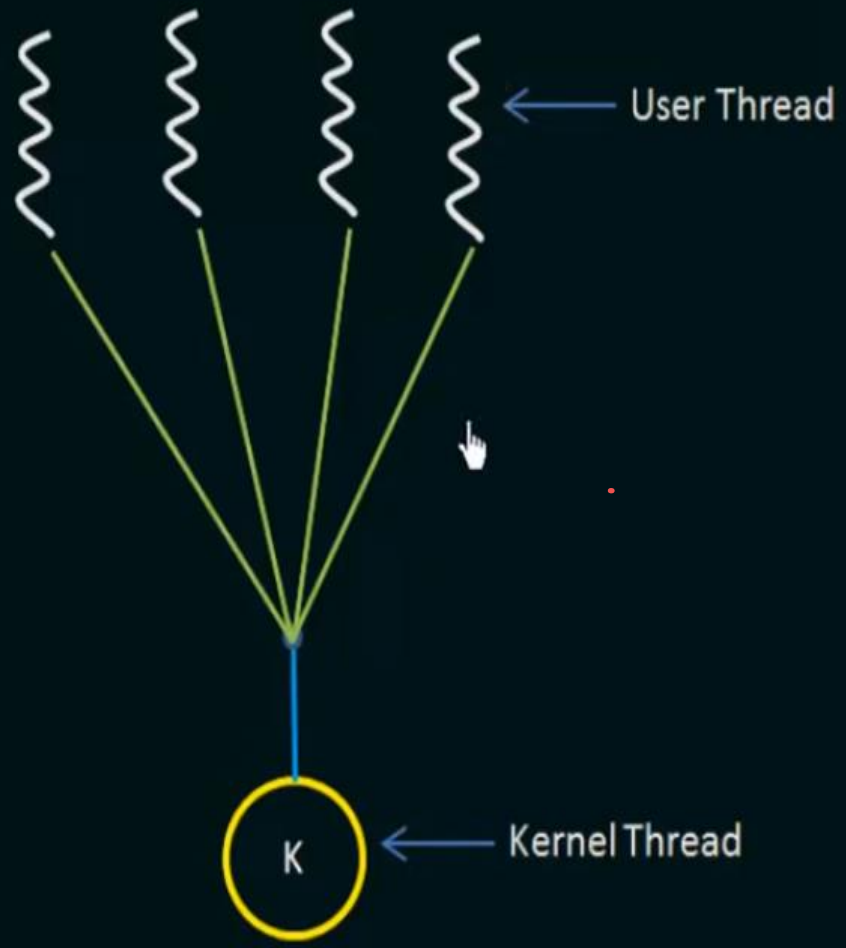
2) Kernel Threads - Supported and managed directly by the operating system.

Ultimately, there must exist a relationship between user threads and kernel threads.

There are three common ways of establishing this relationship:

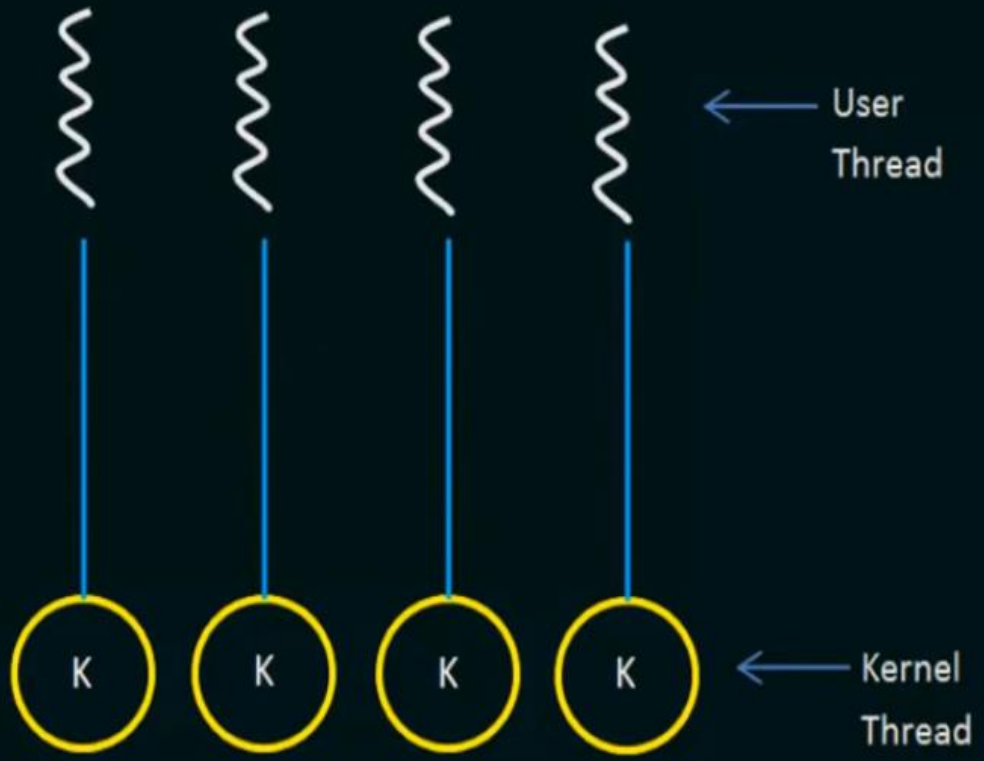
1. Many-to-One Model
2. One-to-One Model
3. Many-to-Many Model

Many-to-One Model



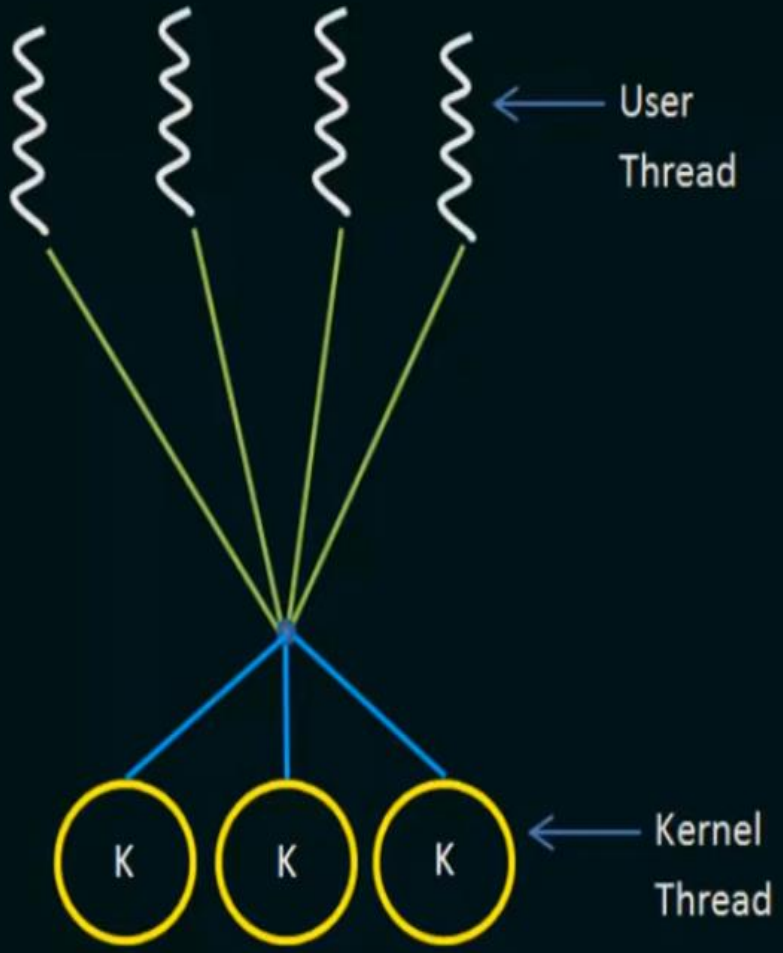
- Maps many user-level threads to one kernel thread.
- Thread management is done by the thread library in user space, so it is efficient.
- The entire process will block if a thread makes a blocking system call.
- Because only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multiprocessors.

One-to-One Model



- Maps each user thread to a kernel thread.
- Provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call;
- Also allows multiple threads to run in parallel on multiprocessors.
- Creating a user thread requires creating the corresponding kernel thread.
- Because the overhead of creating kernel threads can burden the performance of an application, most implementations of this model restrict the number of threads supported by the system.

Many-to-Many Model



- Multiplexes many user-level threads to a smaller or equal number of kernel threads.
- The number of kernel threads may be specific to either a particular application or a particular machine.
- Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.

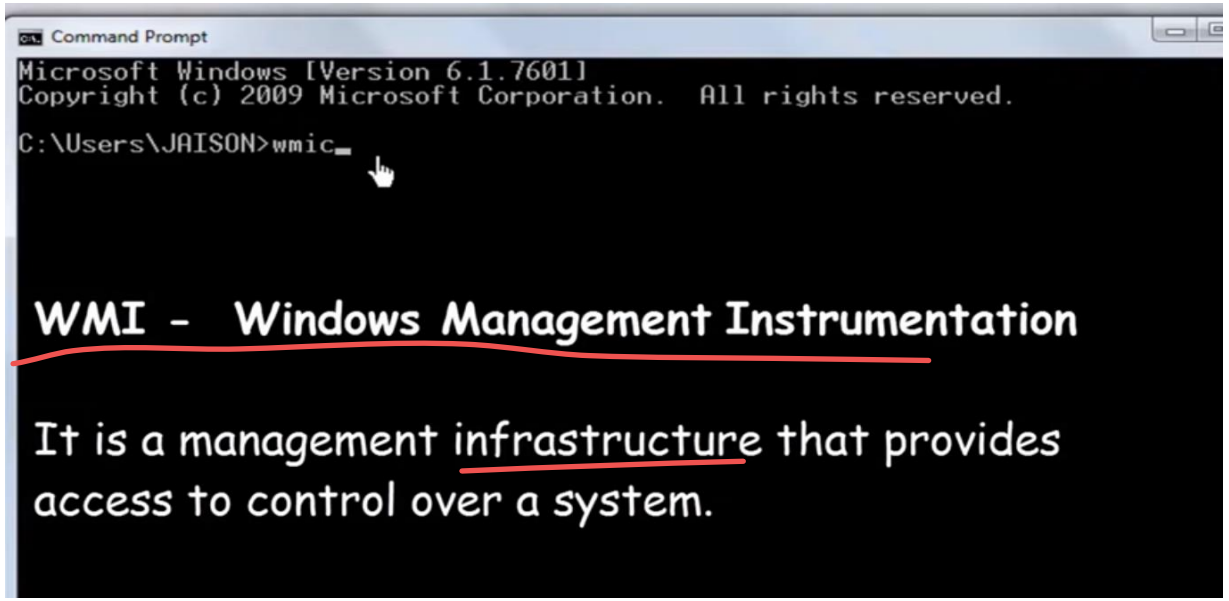
Hyperthreading

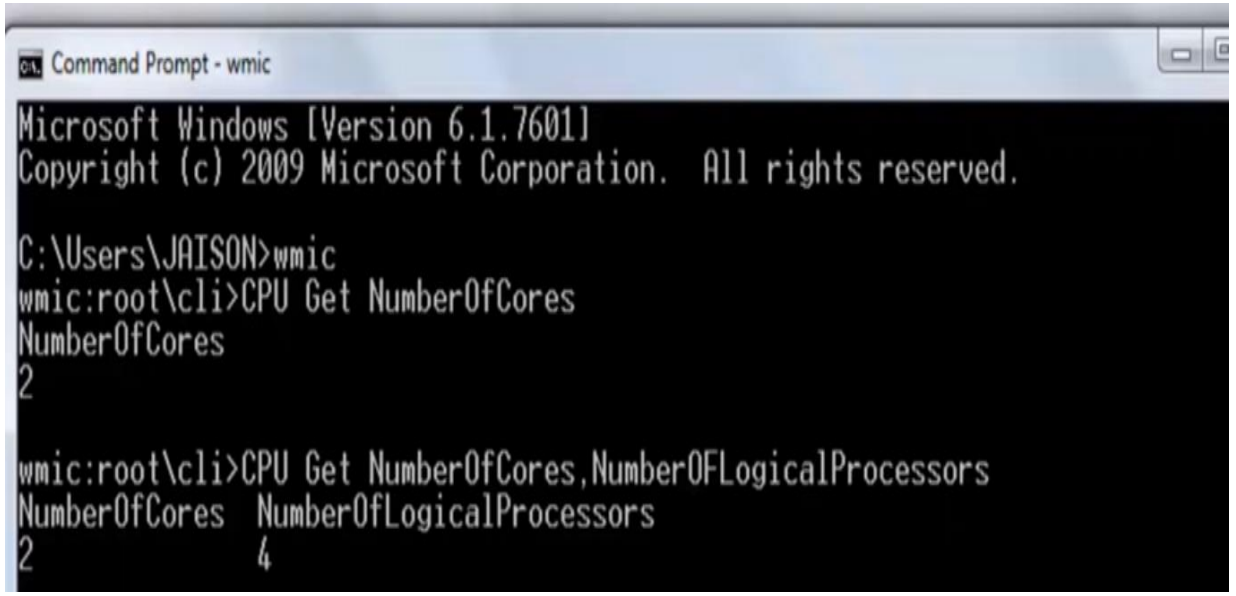
or

Simultaneous Multithreading (SMT)

Hyperthreaded systems allow their processor cores' resources to become multiple logical processors for performance.

It enables the processor to execute two threads, or sets of instructions, at the same time. Since hyper-threading allows two streams to be executed in parallel, it is almost like having two separate processors working together.


Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\JAISON>wmic
WMI - Windows Management Instrumentation
It is a management infrastructure that provides access to control over a system.


Command Prompt - wmic
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\JAISON>wmic
wmic:root\cli>CPU Get NumberOfCores
NumberOfCores
2
wmic:root\cli>CPU Get NumberOfCores,NumberOfLogicalProcessors
NumberOfCores NumberOfLogicalProcessors
2 4