TECHNO INTERNATIONAL BATANAGAR

NAME – SOUMYADEV SANYAL

ROLL- 33200118012

DEPT - COMPUTER SCIENCE ENGINEERING (CSE)

SEMESTER – 5TH

YEAR – 3RD

SUBJECT – OPERATING SYSTEM (OS)

SUBJECT CODE – PCC-CS-502

Term Paper topic – Operating System Multi Threading

Operating System Multi-Threading

What is Thread?

A thread is a flow of execution through the process code, with its own program counter, system registers and

stack. A thread is also called a light weight process. Threads provide a way to improve application performance

through parallelism. Threads represent a software approach to improving performance of operating system by

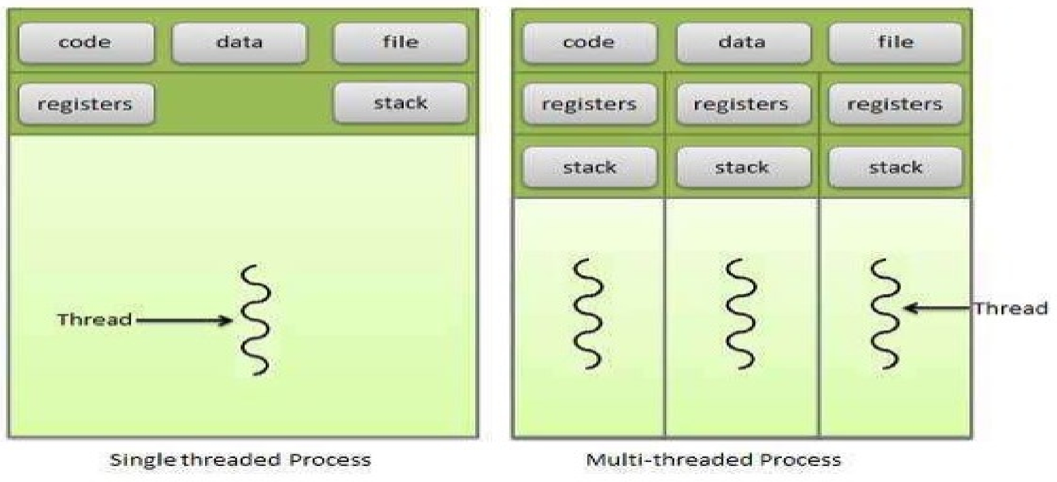
reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a

separate flow of control. Threads have been successfully used in implementing network servers and web server.

They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors.

Following figure shows the working of the single and multithreaded processes.



S.N	PROCESS	THREAD
1	Process is heavy weight or resource intensive.	Thread is light weight taking lesser resources than a process
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.

Advantages of Thread

- Thread minimizes context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Threads

Threads are implemented in following 2 ways.

- User level threads – User managed threads
- Kernel level threads - Operating System managed threads acting on kernel, an operating system core.

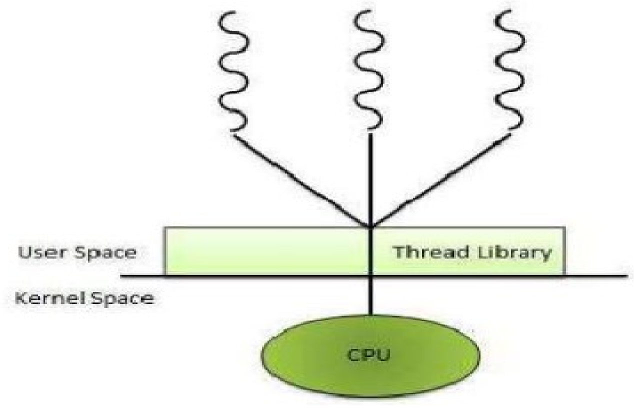
User Level Threads

In this case, application manages thread management kernel is not aware of the existence of threads. The thread

library contains code for creating and destroying threads, for passing message and data between threads, for

scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread

and begins running in that thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical OS, most system calls are blocking.
- Multithreaded applications cannot take advantage of multi-processing.

Kernel Level Threads

In this case, thread management done by the Kernel. There is no thread management code in the application area.

Kernel threads are supported directly by the operating system. Any application can be programmed to be

multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals’ threads within the process.

Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and

management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within same process requires a mode switch to the Kernel.

Multi-Threading

Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good

example of this combined approach. In a combined system, multiple threads within the same application can run in

parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models

are three types

- Many to many relationships.
- Many to one relationship.
- One to one relationship.

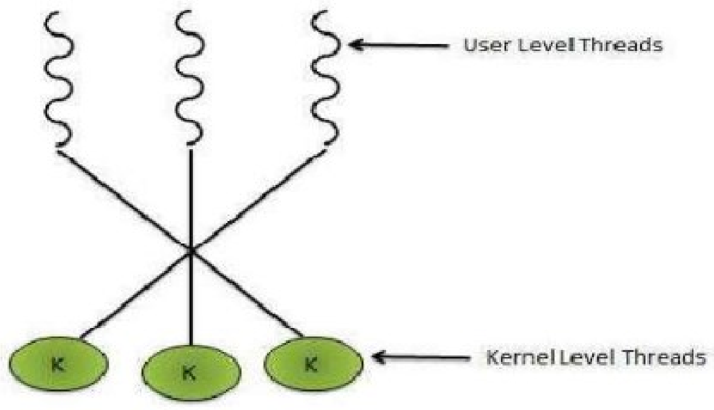
Many to Many Model

In this model, many user level threads multiplexes to the Kernel thread of smaller or equal numbers. The number of

Kernel threads may be specific to either a particular application or a particular machine.

Following diagram shows the many to many model. In this model, developers can create as many user threads as

necessary and the corresponding Kernel threads can run in parallels on a multiprocessor.



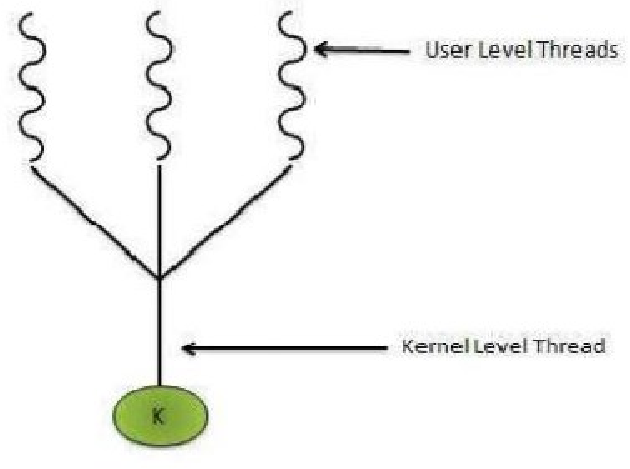
Many to One Model

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in user space. When thread makes a blocking system call, the entire process will be blocks. Only one thread can access the

Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

If the user level thread libraries are implemented in the operating system in such a way that system does not support

them then Kernel threads use the many to one relationship modes.



One to One Model

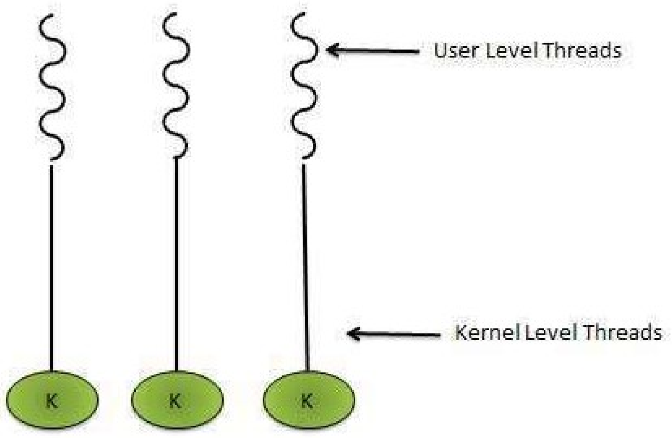
There is one to one relationship of user level thread to the kernel level thread. This model provides more concurrency

than the many to one model. It also another thread to run when a thread makes a blocking system call. It supports

multiple thread to execute in parallel on microprocessors.

Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, Windows NT

and windows 2000 use one to one relationship model.



S.N	**USER LEVEL THREAD**	**KERNEL LEVEL THREAD**
1	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User level thread is generic and can run on any operating system.	Kernel level thread is specific to the operating system.
4	Multi-threaded application cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.