Mawlana Bhashani Science And Technology University

Lab-Report

Report No : 03

Course Code : ICT-3110

Course Title :Operating System Lab.

Date of Performance :

Date of Submission : 13/09/2020

Submitted By:

Name:Md Abdur Rahim

ID:IT-18024

3rd Year 1st Semester

Session : 2017-18

Dept. of ICT

MBSTU

Submitted To:

Nazrul Islam

Assistant Professor

Dept. of ICT

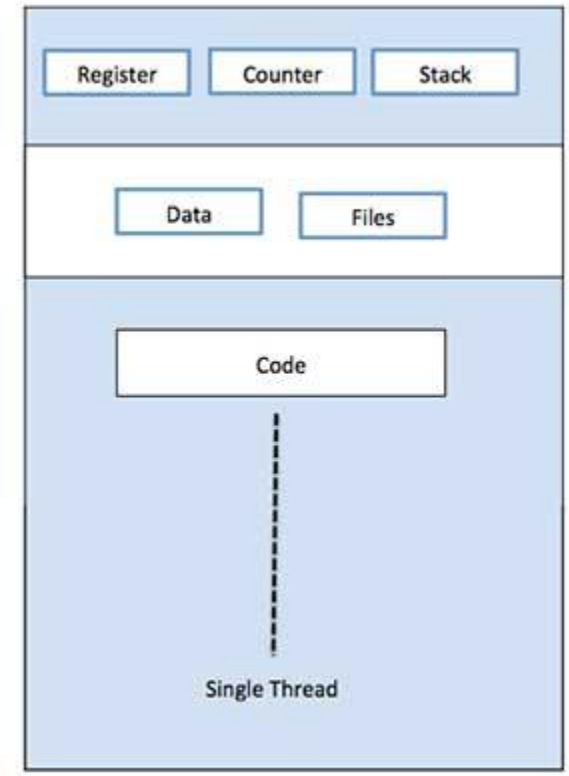
MBSTU

Experiment No : 03

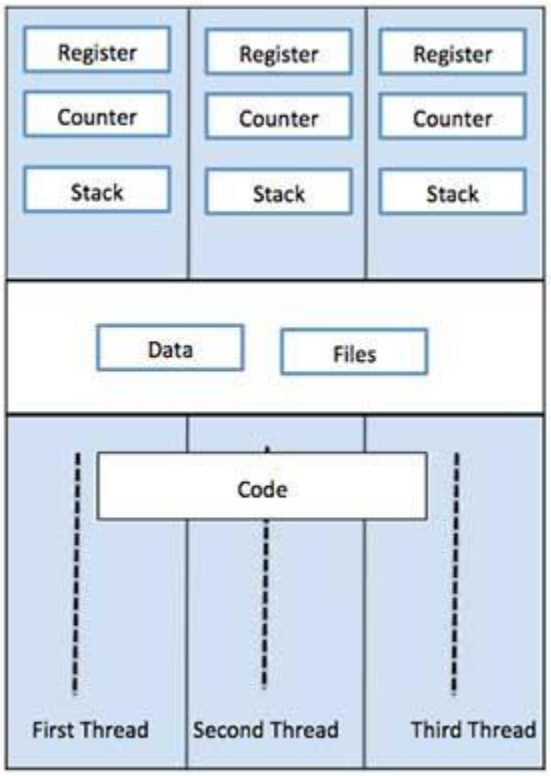
Experiment Name : Threads on Operating System.

Theory : A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history. A thread shares with its peer threads few information like code segment, data segment and open files.

When one thread alters a code segment memory item, all other threads see that. A thread is also called a lightweight 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.



Single Process P with single thread



Single Process P with three threads

Types of Threads:

There are two types of Threads to be managed in Modern systems . They are-

01.User Level Threads.

02. Kernal Level Threads.

User Level Thread : In this 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 starts with a single thread.

- 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.

Kernal Level Thread : Thread management is 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.

- 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 be multithreaded.

Threads Model:

In general, user-level threads can be implemented using one of four models.

- Many-to-one model.
- One-to-one model.
- Many-to-many mode.

• Many-to-one model:

In the many-to-one model all user level threads execute on the same kernel thread. The process can only run one user-level thread at a time because there is only one kernellevel thread associated with the process.

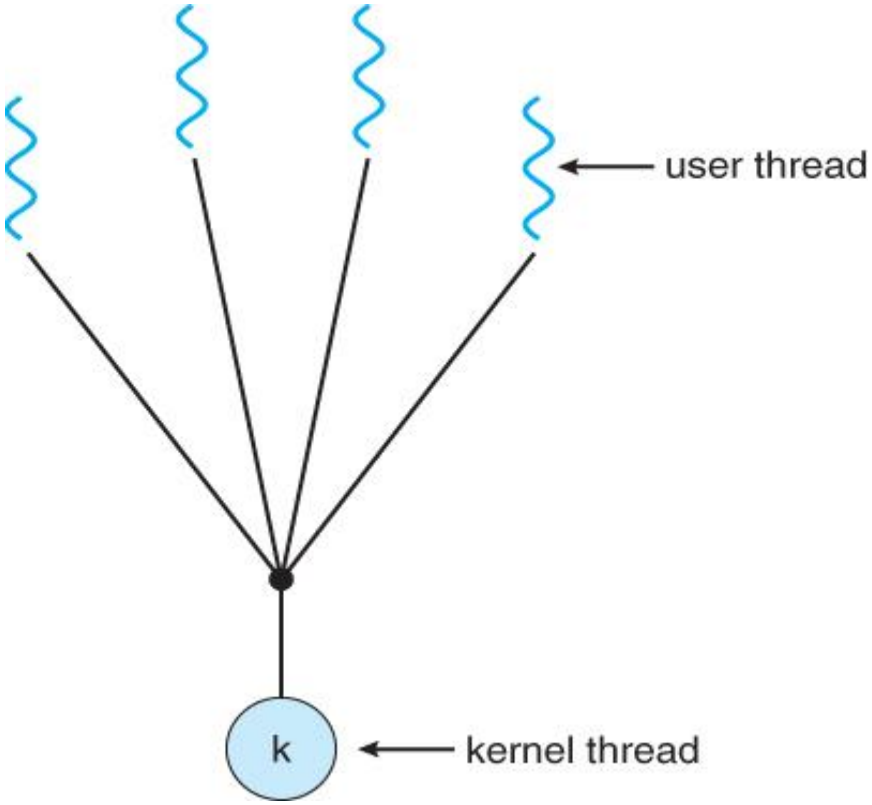


Fig: Many-to-one model

The kernel has no knowledge of user-level threads. From its perspective, a process is an opaque black box that occasionally makes system calls. One to one : In the one-to-one model every user-level thread execute on

One-to-one model:

In the one-to-one model every user-level thread execute on a separate kernel-level thread .The One –to –one model create a separate kernel thread to handle each user thread . One –to-one model overcomes the problems listed above involving blocking system calls and the splitting of processes across multiple CPUs.

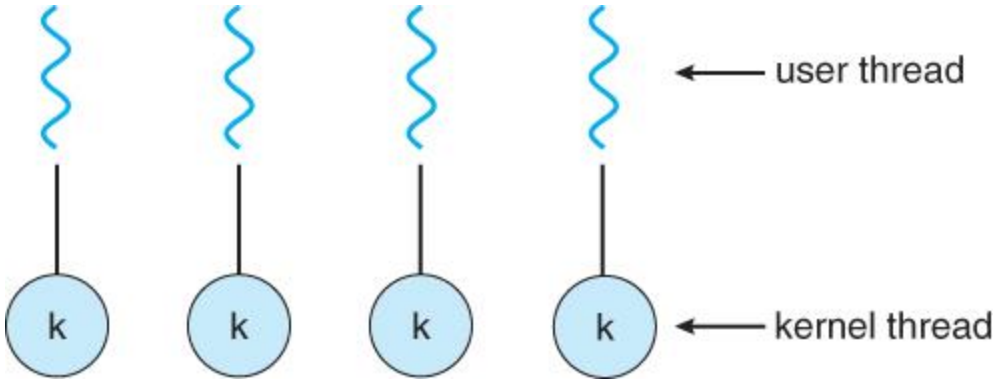


Fig : One –one-model

Many-to-many mode:

In the many-to-many model the process is allocated m number of kernel-level threads to execute n number of user-level thread. The two-level model is similar to the many-to-many model but also allows for certain user-level threads to be bound to a single kernel-level thread.

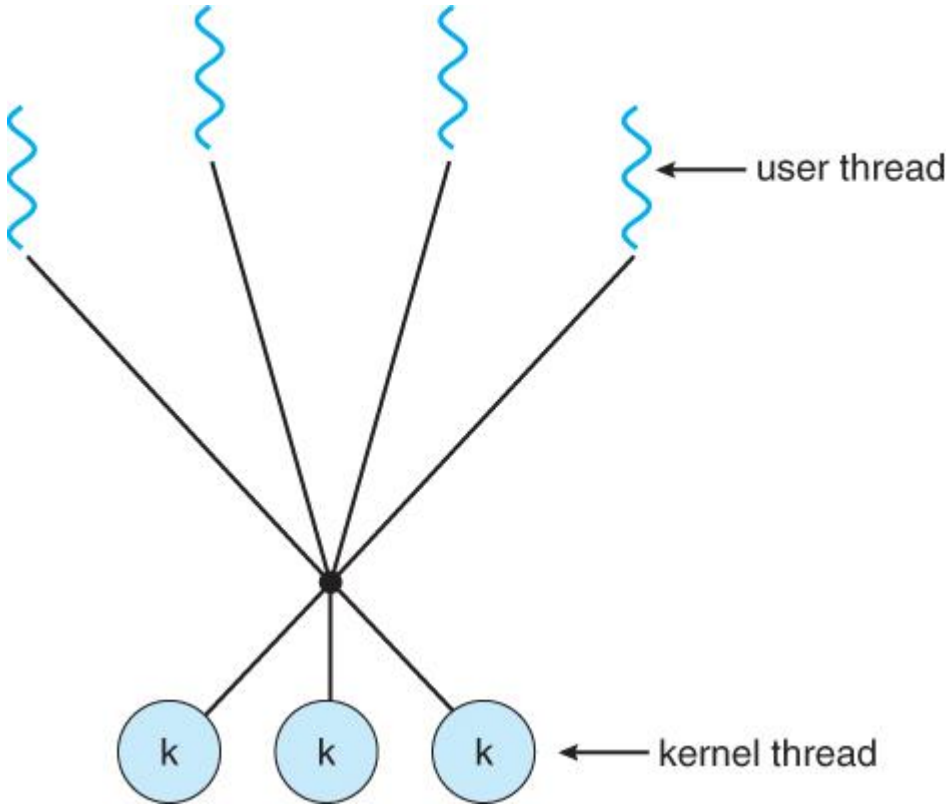


Fig : Many –to-many model

Discussion:

Scalability one thread runs on one CPU. In Multithreaded processes, threads can be distributed over a series of processors to scale. So every operating system thread is so important.