Lab report: 03

Name: Masuk Mia **ID: IT-18049**

i) What is Thread?

Ans: 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.

ii) Types of Threads:

Ans: There are two types of threads. These are given below:

1. User level thread.
2. Kernel level thread.

User Level Threads: The 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.

Advantages:

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 operating system, most system calls are blocking.

Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads : 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.

Advantage:

If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Kernel routines themselves can be multithreaded.

Disadvantages:

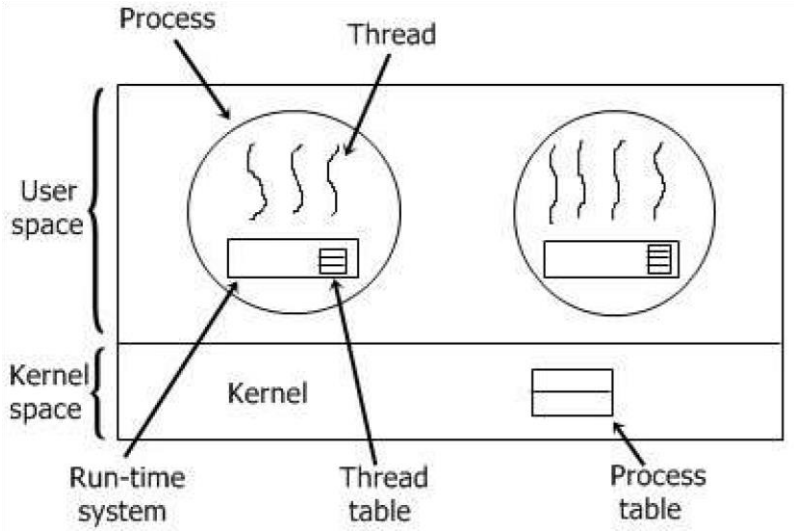
Kernel threads are generally slower to create and manage than the user threads.

Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

iii) Implementation of Threads.

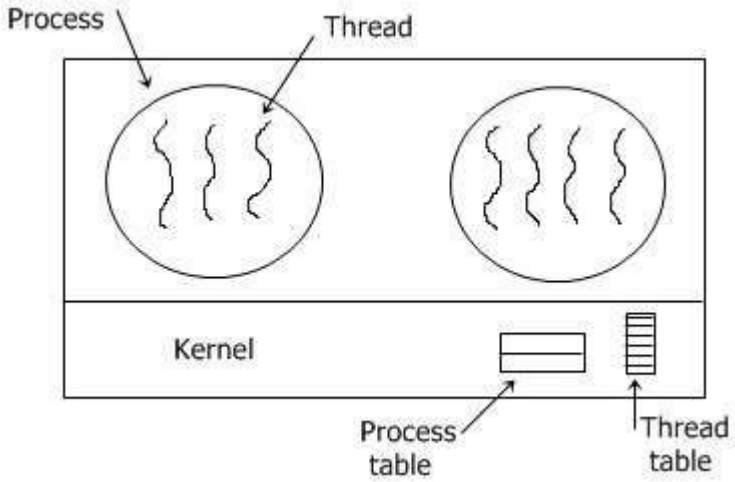
Ans:

Threads Implementation in User Space: In this model of implementing the threads package completely in user space, the kernel don't know anything about them. The advantage of implementing threads package in user space is that a user-level threads package can be implemented on an OS (Operating System) that doesn't support threads. All of these implementations have the same general structure as illustrated in the figure given below.



Threads Implementation in Kernel : In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table. In this method of implementation model, the threads package completely in the kernel. There is no need for any runtime system. To maintain the record of all threads in the system a kernel has a thread table.A call to the kernel is made whenever there is a need to create a new thread or destroy an existing thread. In this, the kernel thread table is updated.

In this method of implementing the threads package entirely in the kernel, no any runtime system is need in each as illustrated in the figure given below.



Now, let's discuss briefly about another two method given here.

- Hybrid implementation
- Scheduler activation

Hybrid Implementation: In this method, each kernel-level thread has some set of userlevel threads that take turns using it.

Scheduler Activation: The goal of this scheduler activation work are to mimic the functionality of kernel threads, but with better performance and greater flexibility generally associated with threads packages implemented in user space.

Conclusion:

From this lab, I can learn how to thread provides concurrency within process and come to learn the concept of thread and its types. Finally I learn about the implementation of threads. User level threads are threads that are visible to the programmer and are unknown to the kernel.