THREADS AND ITS TYPES IN OPERATING SYSTEM

Thread is a single sequence stream within a process. Threads have same properties as of the process so they are called as light weight processes. Threads are executed one after another but gives the illusion as if they are executing in parallel. Each thread has different states.

Each thread has

1. A program-counter
2. A register set
3. A stack-space

Threads are not independent of each other as they share the code, data, OS resources etc.

Similarity between Threads and Processes

1. Only one thread or process is active at a time
2. Within process both execute sequential
3. Both can create children

Differences between Threads and Processes

1. Threads are not independent, processes are.
2. Threads are designed to assist each other, processes may or may not do it

Types of Threads:

1. User Level thread (ULT)
2. Kernel Level Thread (KLT)

User Level thread (ULT)

It is implemented in the user level library; they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.

Advantages of ULT

1. Can be implemented on an OS that doesn't support multithreading.
2. Simple representation since thread has only program counter, register set, stack space.
3. Simple to create since no intervention of kernel.
4. Thread switching is fast since no OS calls need to be made.

Limitations of ULT

1. No or less co-ordination among the threads and Kernel.
2. If one thread causes a page fault, the entire process blocks.

Kernel Level Thread (KLT)

Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition, kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.

Advantages of KLT

1. Since kernel has full knowledge about the threads in the system, scheduler may decide to give more time to processes having large number of threads.
2. Good for applications that frequently block.

Limitations of KLT

1. Slow and inefficient.
2. It requires thread control block so it is an overhead.