Date : 30th - Oct- 2020
Morning Session : 9am – 11.00 PM
By ~ Rohan Kumar

Topics: Operating System -3

Thread in Operating System

What is a Thread?

A thread is a path of execution within a process. A process can contain multiple threads.

Why Multithreading?

A thread is also known as a lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.

Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Advantages of Thread over Process

1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.

3. Effective utilization of a multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processors. This will make process execution faster.

4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.

Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.

5. Communication: Communication between multiple threads is easier, as the threads share common address space. While in process we have to follow some specific communication technique for communication between two processes.

6. Enhanced throughput of the system: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

Types of Threads

There are two types of threads.

User Level Thread

Kernel Level Thread

Threads and its types in Operating System

Thread is a single sequence stream within a process. Threads have the 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 –

- Only one thread or process is active at a time
- Within process both execute sequentially
- Both can create children

Differences between Threads and Processes –

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

Types of Threads:

1. **User Level thread (ULT) –**
   If 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 an 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 –
   - Can be implemented on an OS that does't support multithreading.
   - Simple representation since thread has only program counter, register set, stack space.
   - Simple to create since no intervention of the kernel.
   - Thread switching is fast since no OS calls need to be made.
2. **Disadvantages of ULT –**
   - No or less co-ordination among the threads and Kernel.
   - If one thread causes a page fault, the entire process blocks.
3. **Kernel Level Thread (KLT) –**
   Kernel knows and manages the threads. Instead of a thread table in each process, the kernel itself has a thread table (a master one) that keeps track of all the threads in the system. In addition, the 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 –

- ○ Since the kernel has full knowledge about the threads in the system, schedulers may decide to give more time to processes having large numbers of threads.
- ○ Good for applications that frequently block.

4. **Disadvantages of KLT –**
    - ○ Slow and inefficient.
    - ○ It requires a thread control block so it is an overhead.

Difference between Process and Thread

Process:

Process means any program is in execution. Process control block controls the operation of any process. Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc. A process can create other processes which are known as **Child Processes**. Process takes more time to terminate and it is isolated means it does not share memory with any other process.

Thread:

Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread has 3 states: running, ready, and blocked.

Thread takes less time to terminate as compared to process and like process threads do not isolate.