MULTITREADING AND MULTITASKING

In programming, there are two main ways **to improve the throughput of a program:**

i) **by using multi-threading**

ii) **by using multitasking**

Both these methods take advantage of parallelism to efficiently utilize the power of CPU and improve the throughput of program.

difference between multithreading and multi-tasking

1. The basic difference between multitasking and multithreading is that in multitasking, the **system allows executing multiple programs and tasks at the same time**, whereas, in multithreading, **the system executes multiple threads of the same or different processes at the same time.**
2. Multi-threading is more granular than multi-tasking. In multi-tasking, **CPU switches between multiple programs to complete their execution in real time**, while in multi- threading CPU switches between multiple threads of the same program. Switching between multiple processes has more context switching cost than switching between multiple threads of the same program.
3. Processes are heavyweight as compared to threads. They require their own address space, which means multi-tasking is heavy compared to multithreading.
4. **Multitasking allocates separate memory and resources for each process**/program whereas, in **multithreading threads belonging to the same process shares the same memory and resources as that of the process**.

Comparison between multithreading and multi-tasking

Parameter	Multi tasking	Multi threading
Basic	Multitasking lets CPU to execute multiple tasks at the same time.	Multithreading lets CPU to execute multiple threads of a process simultaneously.
Switching	In multitasking, CPU switches between programs frequently.	In multithreading, CPU switches between the threads frequently.
Memory and Resource	In multitasking, system has to allocate separate memory and resources to each program that CPU is executing.	In multithreading, system has to allocate memory to a process, multiple threads of that process shares the same memory and resources allocated to the process.

Multitasking

Multitasking is when a single CPU performs several tasks (program, process, task, threads) at the same time. To perform multitasking, the CPU switches among these tasks very frequently so that user can interact with each program simultaneously.

In a multitasking operating system, several users can **share the system** simultaneously. CPU rapidly switches among the tasks, so a little time is needed to switch from one user to the next user. This puts an impression on a user that entire computer system is dedicated to him.

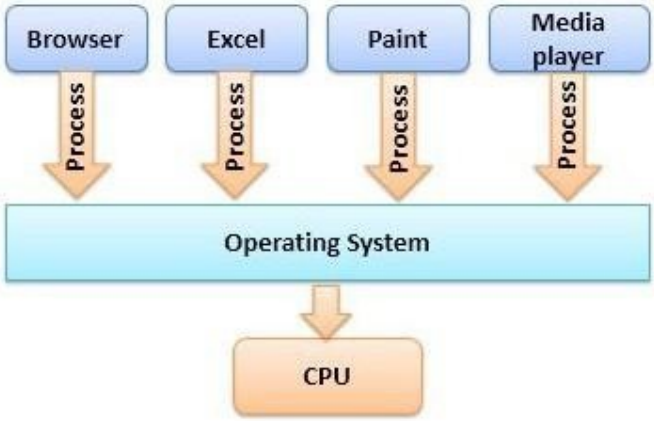

Figure: Multitasking

When several users are sharing a multitasking operating system, CPU scheduling and multiprogramming makes it possible for each user to have at least a small portion of Multitasking OS and let each user have at least one program in the memory for execution.

Multi threading

Multithreading is different from multitasking in a sense that multitasking allows multiple tasks at the same time, whereas, the Multithreading allows multiple threads of a single task (program, process) to be processed by CPU at the same time.

A thread is a basic execution unit which has its own program counter, set of the register and stack. But it shares the code, data, and file of the process to which it belongs. A process can have multiple threads simultaneously, and the CPU switches among these threads so fre- quently making an impression on the user that all threads are running simultaneously.

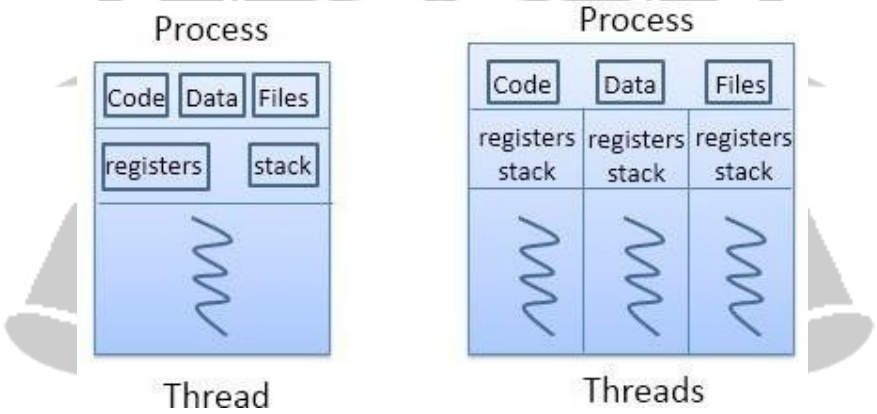

Figure: Multithreading

Benefits of Multithreading

- Multithreading increases the **responsiveness** of system as, if one thread of the

application is not responding, the other would respond in that sense the user would not have to sit idle.

- Multithreading allows **resource sharing** as threads belonging to the same process can share code and data of the process and it allows a process to have multiple threads at the same time active in same address space.
- Creating a different process is costlier as the system has to allocate different memory and resources to each process, but creating threads is easy as it does not require allocating separate memory and resources for threads of the same process.