Create a table to differentiate between threads and processes

Aspect	Threads	Processes
Definition	The smallest unit of execution within a process.	An independent program in execution.
Memory Sharing	Share the same memory space within a process.	Have separate memory spaces.
Creation	Less time-consuming, as they use existing process resources.	More time-consuming, requiring duplication of the parent process.
Communication	Easier and faster through shared memory.	More complex and slower, typically through Inter-Process Communication (IPC) mechanisms like pipes, sockets, or message queues.
Resource Consumption	Lightweight, with minimal overhead.	Heavier, with significant overhead.
Isolation	Less isolated; errors can affect the entire process.	More isolated; errors are less likely to affect other processes.
Concurrency	Enable high concurrency within a single process - while one thread is blocked and waiting, a second thread in the same task can run.	Concurrency across multiple processes - if one process is blocked, then no other process can execute until the first process is unblocked.
Context Switching	Faster, with lower overhead.	Slower, with higher overhead.
Example Usage	Used for tasks requiring high responsiveness, like GUI applications.	Used for executing independent tasks, like running different applications.

Create another table to differentiate between multi-threading and multi-processing

Aspect	Multi-threading	Multi-processing
Definition	Multiple threads within the same process running concurrently.	Multiple processes running concurrently with single-core or in parallel with multi-cores CPU environment.
CPU Cores	Utilizes one CPU core at a time, with threads switching rapidly.	Utilizes multiple CPU cores simultaneously.
Memory Sharing	Threads share the same memory space within a single process. Less memory overhead.	Processes have separate memory spaces. Higher memory consumption.
Communication	Easier and faster through shared memory.	More complex, typically through IPC mechanisms.
Resource Consumption	More efficient, as threads share resources of the parent process.	Less efficient, as each process requires its own resources.
Performance	Better performance in CPU-bound tasks due to lower overhead.	Better performance in I/O-bound tasks due to process isolation.
Fault Isolation	Less fault-tolerant; errors in one thread can affect the entire process.	More fault-tolerant; errors in one process do not affect others.
Context Switching	Faster, with lower overhead.	Slower, with higher overhead.
Synchronization	Requires careful synchronization to avoid race conditions due to shared memory.	Synchronization might be less critical as processes have separate memory spaces.
Concurrency	Achieves tasks execution concurrency	Achieves tasks execution concurrency

	within a single process.	across multiple processes.
Scalability	Limited by Global Interpreter Lock (GIL) in some languages like Python and in cases of single CPU core utilisation.	More scalable, as each process runs independently and in cases of multi-cores CPU utilisation.
Example Usage	Suitable for tasks requiring shared state, like web servers.	Suitable for tasks requiring heavy computation or isolation, like database servers.

Assignment research online references:

https://www.guru99.com/cpu-core-multicore-thread.html#6
https://workat.tech/core-cs/tutorial/processes-and-threads-os-6iboki1s2y3t
https://www.guru99.com/cpu-core-multicore-thread.html
https://www.tutorialspoint.com/operating_system/os_memory_management.htm
https://towardsdatascience.com/multithreading-and-multiprocessing-in-10-minutes-20d9b3c6a867