Parallel processing vs threads

Parallel Processing:

- Parallel processing involves using multiple processors or cores to execute multiple tasks or parts of a task simultaneously.
- It's a higher-level concept that refers to the overall approach of breaking down a task into smaller subtasks that can be processed concurrently.
- Parallel processing is often used to tackle computationally intensive tasks that can be divided into smaller, independent pieces.
- It can be used in various scenarios, such as scientific simulations, rendering, data analysis, and more.
- Examples of parallel processing models include SIMD (Single Instruction, Multiple Data) and MIMD (Multiple Instruction, Multiple Data) architectures.

Threads:

- Threads are the smallest unit of a program that can be scheduled for execution. They share the same memory space within a process and can communicate and synchronize with each other more easily compared to separate processes.
- Threads within a process share resources like memory, file descriptors, and code segments.
- Threads can be implemented in both single-core and multi-core systems.
- Threads are more lightweight than processes and have less overhead when it comes to context switching and communication.
- Threads can be used for multitasking within a single process, allowing different parts of the program to run concurrently.