Was ist Multithreading?

Unter einem **Hardware-Thread** versteht man eine Hardware-Ausführungseinheit, die ein Programm abarbeiten kann. Dazu gehören CPU-Register, CPU-Recheneinheiten usw. Moderne CPUs besitzen mehrere Hardware-Threads (derzeit 4-32), Server CPUs sogar sehr viele (derzeit bis 128), sodass man eine große Anzahl von Programmen gleichzeitig ausführen kann.

Unter einem **Software-Thread** versteht man eine Software-Struktur, die ein ganzes Programm (z.B. in Python, C++) oder auch nur einen Teil eines Programmes (Funktion, Lambda) ausführen kann. Dazu gehören reservierte Speicherbereiche (Stack, Static Storage…) und dazugehörige Verwaltungsstrukturen. Ein „normales“ (= singlethreaded) Programm hat nur einen einzigen Software-Thread, der zu jedem Zeitpunkt nur von einem einzigen Hardware-Thread ausgeführt wird. Ein multithreaded Programm besteht aus mehreren (oder sehr vielen) Software-Threads, die auf verschiedenen Hardware-Threads parallel laufen können (wodurch die Ausführungszeit hoffentlich kleiner wird) oder auf einem einzigen Hardware-Thread abwechselnd ausgeführt werden (wodurch sich natürlich keine Beschleunigung ergibt). Die Entscheidung, ob und wie viele Hardware-Threads für ein Programm verwendet werden, trifft das Betriebssystem in Abhängigkeit von der Hardware und der Systemauslastung.

Um vorhandene Hardware-Threads zu nutzen, muss man das Programm (oder einen Teil davon) immer in Software-Threads aufteilen. Das sind C++Funktionen (oder Methoden oder Lambdas), die einen Teil der Arbeit übernehmen. Der Haupt-Thread (bei singlethreaded Programmen ist das auch der einzige Thread) arbeitet die Funktion `main()` ab und wird bei der Programmausführung vom Betriebssystem erzeugt, die weiteren Threads müssen vom Programm oder von den verwendeten Bibliotheken erzeugt werden.

Multithreading macht nur Sinn, wenn

1) CPUs mit genügend Hardware-Threads eingesetzt werden können.
2) das Programm eine nennenswerte Laufzeit hat (mindestens 10 Sekunden).
3) das Programm leicht in einzelne Teilabschnitte zerlegbar ist.
4) die Koordination der Software-Threads nicht mehr Zeit kostet als man durch die mögliche Parallelität gewinnt.
5) man ausreichend leidensfähig ist, denn Multithreading ist keineswegs einfach und man hat hier deutlich mehr Fehlerquellen zu beachten.

Die C++-Threadbibliothek ist noch relativ klein, mit jedem neuen Sprachstandard kommen neue Features dazu (z.B. parallele STL-Algorithmen und `jthreads` in C++20).