Guarded Suspension

A guarded suspension consists of a lock and a condition, which has to be fulfilled by the calling thread.

- The calling thread will put itself to sleep if the condition is not meet.
- The calling thread uses a lock when it checks the condition.
- The lock protects the calling thread from a data race or deadlock.

Guarded Suspension can vary in many ways:

- The waiting thread is passively informed about the state change or actively requests it.
  Push versus Pull Principle
- The waiting thread can wait with or without a time limit.
- The notification can be sent to one or all waiting threads.

Guarded Suspension

- Guarded suspension is in C++ available in many variations.
  - The waiting thread is notified about the state change or asks for the state change.
    - Push principle: condition variables, future/promise pairs, atomics (C++20), or semaphores (C++20)
    - Pull principle: not natively supported in C++
  - The acquiring thread tries it only once.
    - Not natively supported but can be implemented using condition variables, future/promise pairs, or atomics (C++20)
  - The waiting thread waits with or without a time limit.
    - Condition variables, or future/promise pairs
  - The notification is sent to one or all waiting threads.
    - Shared futures, condition variables, atomics (C++20), or semaphores (C++20)