

# Non-real-time Mutate Summary

- Scenario:
  - Data is big: `std::atomic<>::is_always_lock_free == false`
  - The non-real-time thread **can** mutate the object
  - Real-time thread will not fail to acquire the resource
- Trade-off:
  - The real-time thread **can not** mutate the object
  - Non-real-time thread will wait on the real-time thread
  - Overhead of copying on the non-real-time thread
- Examples:
  - Sharing large data from the non-real-time thread to the real-time thread
  - Audio samples, wavetables, filter coefficients etc.

# Real-time Mutating?

