Semaphore vs Monitor

This lesson discusses the differences between a monitor and a semaphore.

Semaphore vs Monitor

Monitor, mutex, and semaphores can be confusing concepts initially. A monitor is made up of a mutex and a condition variable. One can think of a mutex as a subset of a monitor. Differences between a monitor and a semaphore are discussed below.

The Difference

- A monitor and a semaphore are interchangeable. Theoretically, one can be constructed out of the other or one can be reduced to the other. However, monitors take care of atomically acquiring the necessary locks, whereas, with semaphores, the onus of appropriately acquiring and releasing locks is on the developer, which can be error-prone.

- Semaphores are lightweight when compared to monitors, which are bloated. However, the tendency to misuse semaphores is far greater than monitors. When using a semaphore and mutex pair as an alternative to a monitor, it is easy to lock the wrong mutex or just forget to do it altogether. Even though both constructs can be used to solve the same problem, monitors provide a pre-packaged solution with less dependency on a developer's skill to get the locking right.

- In Ruby, the `Monitor` class is the implementation of the monitor concept. And a condition variable associated with a monitor object can be created using the method `new_cond()`. Condition variables enforce correct locking by raising an exception when a thread attempts to invoke the `wait()` or `notify()` methods without entering attempts to invoke the `wait()` or `notify()` methods without entering the monitor associated with the condition variable.

- A semaphore can allow several threads access to a given resource or critical section. However, only a single thread can **own** the monitor and access associated resource at any point.
- Semaphores can be used to address the issue of **missed signals**. However, with monitors, the additional state, called the predicate, needs to be maintained apart from the condition variable and the mutex, which make up the monitor, to solve the issue of missed signals.