**Apa perbedaan antara monitor dan semaphore?**

|  |
| --- |
|  |

Alasan bahwa Semaphore dan monitor diperlukan karena aplikasi multi-threaded (seperti Microsoft Word, Excel, dll) harus mengendalikan bagaimana thread mengakses sumber daya bersama. Hal ini dikenal sebagai **sinkronisasi thread** - yang benar-benar diperlukan dalam aplikasi multi-threaded untuk memastikan bahwa thread bekerja dengan baik dengan satu sama lain. Jika aplikasi tidak mengontrol thread maka dapat mengakibatkan korupsi data dan masalah lainnya.

## Apakah saya menggunakan monitor atau semaphore?

Monitor dan Semaphore keduanya konstruksi pemrograman yang digunakan untuk mencapai sinkronisasi thread.   
Apakah Anda harus menggunakan monitor atau semaphore? itu tergantung pada bahasa atau sistem yang mendukung.

## Apa itu Monitor?

|  |
| --- |
|  |

Sebuah monitor adalah **seperangkat** rutinitas yang dilindungi oleh mutual exclusion lock. Tidak satu pun dari rutinitas di monitor dapat dieksekusi oleh thread sampai thread memperoleh kunci. artinya hanya satu thread yang dapat mengeksekusi dalam monitor pada suatu waktu. Setiap thread lain harus menunggu thread yang sedang dijalankan untuk menyerahkan kontrol kunci.

Namun, thread dapat menangguhkan dirinya di dalam monitor dan kemudian menunggu untuk sebuah event terjadi. Jika event tersebut terjadi, maka thread lain diberi kesempatan untuk masuk monitor. Thread yang ditangguhkan pada akhirnya akan diberitahu bahwa event itu sedang menunggu untuk diisi, yang berarti dapat kembali dan mengambil kunci untuk di eksekusi.

.

## Perbedaan antara Monitor dan Semaphore

|  |
| --- |
|  |

Monitor dan Semaphore digunakan untuk tujuan yang sama - sinkronisasi thread. Tapi, monitor sangat sederhana untuk digunakan dibandingkan Semaphore karena mereka menangani semua rincian akuisisi kunci dan rilis. Aplikasi yang menggunakan Semaphore harus melepaskan kunci thread yang telah diperoleh ketika aplikasi berakhir, hal ini harus dilakukan oleh aplikasi itu sendiri. Jika aplikasi tidak melakukan hal ini, maka setiap thread lain yang membutuhkan sumber daya bersama tidak akan dapat melanjutkan.

Perbedaan lain ketika menggunakan Semaphore adalah bahwa setiap rutinitas yang mengakses sumber daya bersama harus secara eksplisit mendapatkan kunci sebelum menggunakan sumber daya. Hal ini dapat dengan mudah dilupakan ketika coding rutinitas berurusan dengan multithreading. Monitor tidak seperti Semaphore, otomatis memperoleh kunci yang diperlukan.

## Apakah ada biaya untuk menggunakan monitor atau semaphore?

Ya, ada biaya yang terkait dengan penggunaan konstruksi sinkronisasi seperti monitor dan Semaphore. Dan, biaya ini adalah waktu yang diperlukan untuk mendapatkan kunci yang diperlukan setiap kali sumber daya bersama diakses.