**Monitor dibagi kedalam tiga jenis**, berikut adalah contoh-contoh monitor yaitu :

A. Monitor dengan Signal (Monitor Hoare)

Monitor ini mempunyai prosedur didalam modul monitor dimana hanya dapat diakses oleh prosedur lokal, jika suatu proses masuk kedalam monitor yaitu dengan cara meminta salah satu prosedur yang terdapat pada monitor. Suatu saat hanya terdapat satu proses yang dapat dieksekusi dalam monitor(proses lainnya menunggu giliran dan tidak membutuhkan semaphore). Jika terdapat data variabel global maka akan dapat dilindungi bila ditempatkan didalam monitor.

B. Monitor dengan Notify (Monitor Lampson-Redell)

Adapun tujuan dari Monitor dengan Notify adalah untuk mengatasi kelemahan yang terdapat di monitor. Pada cnotify berfungsi untuk memberitahu kondisi dari proses di dalam antrian (misal : not empty, not full, dll). Proses dalam antrian tersebut tidak harus segera dieksekusi, namun menunggu hingga monitor dalam kondisi tidak sibuk, maka diperlukan pengecekan status monitor (looping).

C. Monitor dengan Broadcast (Monitor Lampson-Redell)

Monitor dengan Broadcast merupakan pengembangan daripada monitor sebelumnya yaitu Monitor dengan Notify. Cbroadcast digunakan pada kondisi dimana jumlah proses yang harus diaktifkan tidak diketahui secara pasti, cbroadcast(x) akan membuat seluruh proses didalam antrian akan diubah statusnya menjadi ready ketika menunggu eksekusi kondisi x.

**Implementasi dari Mutual Exclusion dengan Semaphore**

1. Pemanfaatan semaphore primitif dalam mutex

Contoh dari Pemanfaatan semaphore primitif dalam mutex yaitu misal terdapat dua buah thread yang sedang berjalan bersamaan :

thread A :                         thread B :

count = count +1;          count = count + 1;

karena thread A dan thread B mengakses variabel yang sama (count) sehingga thread A dan thread B diharuskan berjalan satu-satu dengan cara menggunakan semaphore mutex yang berupa binary semaphore dengan nilai awal 1.

thread A :                         thread B :

kunci(mutex);                kunci(mutex);

count = count + 1;         count = count + 1;

buka(mutex);                 buka(mutex);

thread manapun yang mengeksekusi kunci terlebih dahulu akan jalan terus, sedangkan untuk thread yang tiba belakangan akan menunggu sampai thread yang sudah berjalan terlebih dahulu mengeksekusi ‘buka’, setelah itu kedua thread berjalan lagi dengan normal.

2. Urutan eksekusi tiga buah proses dengan semaphore

Terdapat tiga buah thread yang berjalan bersamaan. Resource yang tersedia hanya cukup untuk dua buah thread.

thread A :                         thread B :                            thread C :

//critical section             //critical section                //critical section

Harus dilakukan pengaturan karena pada suatu saat hanya ada dua buah thread yang berada pada critical sectionnya, dengan menggunakan semaphore multiplex yaitu sebuah counting semaphore dengan nilai awal sama denga jumlah resource yang tersedia, yaitu dua.

thread A :                         thread B :                            thread C :

kunci(mutex);                kunci(mutex);                   kunci(mutex);

//critical section           //critical section              //critical section

buka(mutex);                 buka(mutex);                    buka(mutex);

jika dua buah thread sedang berada dalam critical section, thread berikutnya yang akan memasuki critical section harus menunggu kedua thread tersebut selesai dieksekusi untuk dapat memasuki critical sectionnya.