Systèmes d'exploitation - Moniteurs II

Othmane AJDOR

2018-2019

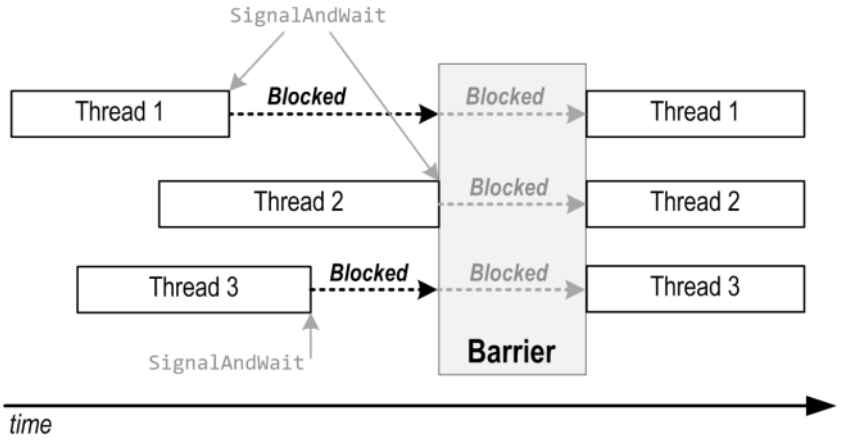
1 But

— Synchronisation de threads, 1 même programme , mémoire partagée
— Ensemble de fonctions en exclusion mutuelle (pas de problème de concurrence)

Le problème est de prendre des decisions alors qu'on est en exclusion mutuelle. On empeche le programme d'évoluer (mise en attente à l'aide de variables de condition), ainsi, on relache l'exclusion mutuelle et les threads sont placés dans une file d'attente.

2 Barrière à N

On veut que N threads appellent barriereN()



On utilise un mutex[lock|unlock], la fonction cnd_t[wait|signal].
cnd_signal(&c) réveille un thread.

```
int compteur = 0;
const N = 42;
mtx_t m;
cnd_t c;
barriereN(){
        mtx_lock();
        compteur++;
        while(compteur < N){
                cnd_wait(&c, &m);
        }
        // reveil en cascade, le N+1 reveil le premier et ainsi de suite
        cnd_signal(&c);
        mtx_unlock();
}
```

3 Allocateur

On se contente de reveiller tous les threads en attente.
cnd_broadcast(&c) reveille tous les threads.

```
const int resource = N;
mtx_t m;
cnd_t c;
allocation(int n){
        mtx_lock(&m);
        while(resource<n){
                cnd_wait(&c, &m);
        }
        resource -= n;
        mtx_unlock(&m);
}

liberation(int n){
        mtx_lock(&m);
        resource += n;
        cnd_broadcast(&c);
        mtx_unlock(&m);
}
```