Moniteurs

PCO
7 - Moniteurs

Résumé du document
Definition

Table des matières

1. Variable de condition

Une variable de condition est une variable qui permet de synchroniser l'exécution de plusieurs processus. Elle est associée à un moniteur et permet de bloquer un processus en attente d'une condition.

Dans le cadre du cours on parle alors du type `Condition` et propose deux opérations:

- `wait` : qui permet de
  - bloquer inconditionnellement la tâche appelante
  - lui fait relâcher l'exclusion mutuelle sur le moniteur
  - la place dans la liste d'attente de la condition
- `signal` : qui permet de
  - dépend de l'état de la file associée à la condition
  - si la file est **vide** alors rien ne se passe
  - si la file est **non vide** alors un processus est débloqué et reprend immétaitement l'exécution

2. Moniteur de Mesa

Dans un moniteur de Mesa, les variables de conditions sont utilisées pour synchroniser les processus. Pour ce type de moniteur nous devons systématiquement avoir:

- un `mutex` qui assure l'exclusion mutelle
- une `condition` de **type** `PcoConditionVariable`

2.1. Fonctions

2.1.1. Attente

La fonction `wait` permet de bloquer un processus en attente d'une condition. Dans le cas ou nous utilisons la fonction `wait` nous aurions les étapes suivantes:

```
void PcoConditionVariable::wait(PcoMutex * lockedMutex)
```

- de manière **atomique** on effectue:
  - relâcher le mutex
  - attendre que la condition soit signalée
- l'exécution du thread est suspendue jusqu'à que la condition soit signalée
- le mutex doit être **verrouillé** par le thread avant d'appeler la fonction `wait()`
- au moment où la condition est **signlée**, `wait()` re-verouille automatiquement le mutex

Attention, lors du re-verouillage le thread est en compétition avec tous les threads bloquant sur le verrou!

Pour éviter tout problème de **deadlock** il est plus que recommandé de toujours appeler la fonction `wait()` dans une boucle `while` qui teste la condition.

2.1.2. Signalisation

2.1.2.1. `notifyOne`

La fonction `notifyOne` permet de signaler un processus en attente d'une condition. Dans le cas ou nous utilisons la fonction `notifyOne` nous aurions les étapes suivantes:

```
void PcoConditionVariable::notifyOne();
```

- réveille **un** des threads en attente sur la variable de condition
  - si aucun thread n'est en attente, la fonction ne fait rien
  - si plusieurs threads sont en attente, un thread est choisi arbitrairement
    - **Pas** de liste d'attente FIFO, choix par l'ordonnanceur

2.1.2.2. `notifyAll`

La fonction `notifyAll` permet de signaler tous les processus en attente d'une condition. Dans le cas ou nous utilisons la fonction `notifyAll` nous aurions les étapes suivantes:

```
void PcoConditionVariable::notifyAll();
```

- réveille **tous** les threads en attente sur la variable de condition
  - si aucun thread n'est en attente, la fonction ne fait rien
  - si plusieurs threads sont en attente, tous les threads sont réveillés mais un seul à la fois peut acquérir le mutex (ordre imprévisible, dépend de l'ordonnanceur)

2.2. Exemple

```
class Buffer {
private:
    PcoMutex mutex;
    PcoConditionVariable isFree, isFull;
    std::string buffer;
    bool bufferFull;

public:
    Buffer() : bufferFull(false) {}

    void put(std::string msg);
    std::string get(void);
};
```

```
// Dépose le message msg (qui est dupliqué)
// et bloque tant que le tampon est plein.
void Buffer::put(std::string msg) {
    mutex.lock();
    while (bufferFull)
        isFree.wait(&mutex);
    buffer = msg;
    bufferFull = true;
    isFull.notifyOne();
    mutex.unlock();
}
```

```
// Renvoie le message du tampon et bloque
// tant que le tampon est vide.
std::string Buffer::get() {
    std::string result;
    mutex.lock();
    while (!bufferFull)
        isFull.wait(&mutex);
    result = buffer;
    bufferFull = false;
    mutex.unlock();
    isFree.notifyOne();
    return result;
}
```

3. Moniteur de Hoare

Le moniteur de Hoare est semblable au moniteur de Mesa, mais avec quelques différences. Dans un moniteur de Hoare:

- plus besoin de gérer de mutex