Mutex et moniteurs

vendredi 27 octobre 2017 10:16

Question 1

```
int value = 0;

void retirer(int n) {
  value -= n;
}

void ajouter(int n) {
  value += n;
}
```

Question 2

```
int value = 0;
mutex m;

void retirer(int n) {
  m.lock();
  value -= n;
  m.unlock();
}

void ajouter(int n) {
  m.lock();
  value += n;
  m.unlock();
}
```

Question 3

```
int nbThreads = 0;
mutex m;
cond f;

void barrier() {
  m.lock()
  nbThreads++;
  while (nbThreads < N)
    f.wait()
  f.broadcast();
  m.unlock();
}
```

Le processus pouvant se réveiller sans signalisation, on utilise un `while` avant le `wait` !

Question 4

```
const int N = 100; //cases dans le tampon
Message tampon[N]; //buffer d'écriture
mutex m;
cond vide;
cond plein;
int count = 0;
int iprod = 0;
int icons = 0;

void producteur(Message mes)
{
  m.lock();
```

```
  while (count == N)
    plein.wait()
  tampon[iprod] = Mess
  iprod = (iprod + 1)%N;
  count ++;
  vide.signal();
  m.unlock();
}

void consommateur(Message *mes)
{
  m.lock();
  while (count == 0)
    vide.wait()
  *mes = tampon[icons]
  icons = (icons + I)%N;
  count --;
  plein.signal();
  m.unlock();
}
```

Question 5

Oui

Question 6

Attention : il faut locker le mutex avant d'arriver sur la condition.

`signal` : réveille la dernière condition
`broadcast` : réveille tout le monde

```
mutex m;
int nb_lect = 0;
bool ecriture = false;
cond lect;
cond redac;
void debut_lire() {
    m.lock();
    nb_lect ++;
    while (ecriture)
        lect.wait();
    m.unlock();
}
void fin_lire() {
    m.lock();
    nb_lect --;
    if (nb_lect == 0)
        redac.signal();
    m.unlock();
}
void debut_redac() {
    m.lock();
    ecriture = true;
    while (ecriture || nb_lect > 0)
        redac.wait();
    m.unlock();
}
void fin_redac() {
    m.lock();
    ecriture = false;
    if (nb_lect > 0)
```

```
        lect.broadcast();
    else
        redac.signal();
    m.unlock();
}
```

Question 7

Problème de la priorité aux lecteurs : s'il y a toujours quelqu'un en lecture, on ne pourra jamais écrire !

Question 8

```
mutex m;
int nb_lect = 0;
int nb_lect_attente = 0;
int nb_redac = 0;
int nb_redac_attente = 0;
bool ecriture = false;
cond lect;
cond redac;
void debut_lire() {
    m.lock();
    nb_lect_attente ++;
    while (ecriture || nb_redac > 0)
        lect.wait();
    nb_lect_attente --;
    nb_lect ++;
    signal(lect); // Réveil en cascade
    m.unlock();
}
void fin_lire() {
    m.lock();
```

```
    nb_lect --;
    if (nb_lect == 0 &&
nb_redac_attente >0)
        redac.signal();
    m.unlock();
}
void debut_redac() {
    m.lock();
    nb_redac_attente ++;
    while (nb_lect > 0 || ecriture)
        redac.wait();
    ecriture = true;
    nb_redac_attente --;
    m.unlock();
}
void fin_redac() {
    lock();
    ecriture = false;
    if (nb_redac_attente > 0)
        redac.signal();
    else if (nb_lect_attente > 0)
        lect.signal();
    m.unlock();
}
```

Question 9

```
int nblibre = N;
bool alloc(int n) {
    if (nblibre < n)
        return false;
    nblibre -= n;
    return true;
```

```
}
void free(int n) {
    nblibre += n;
}
```

Question 10