Chapitre 3
Coopération et synchronisation par variables partagées

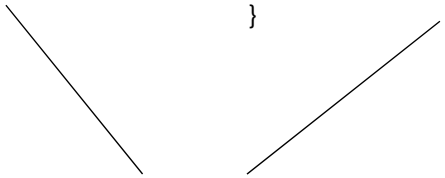
Problème de l'exclusion mutuelle
action atomique
Protocoles d'exclusion mutuelle

Sémaphores

3.1. Problème de l'exclusion mutuelle

Problème : Deux processus cycliques partageant une imprimante.
Chaque processus imprime son identité n fois.
Nous souhaitons que les résultats de l'impression ne soient pas mélangés

```
P0                                  P1
{  for(;;)                          {  for(;;)
        imprimer "p0" n fois;               imprimer "p1" n fois;
}                                   }
```



Rien n'empêche d'imprimer une séquence P0P0P1P0P1….

Comment empêcher ce phénomène?

Comment exécuter la section critique en exclusion mutuelle

3.2 Action atomique

Considérons un système centralisé

Action atomique: action indivisible par exemple une instruction câblée

Considérons un langage de programmation simple du style C
- **séquence, if, while..**
- **les processus peuvent utiliser des variables partagées (VP)**
- **affectation d'une VP par une valeur est atomique**
- **comparaison d'une VP avec une valeur est atomique**
- **pas d'instructions spéciales dans ce langage**

3.3 Protocoles d'exclusion mutuelle

- Nous allons construire un protocole de l 'exclusion mutuelle entre deux processus.
- Nous considérons deux processus cycliques dénotés P0 et P1 qui partagent une section critique dénotées SC0 dans P0 et SC1dans P1.

Nous prenons en compte les hypothèses suivantes:

H1. Un processus ne restent pas infiniment dans sa section critique

H2. Un processus en dehors du protocole ne peut pas interdire l'accès à la section critique

H3. le protocole est construit en utilisant (un minimum) de variables partagées.

H4. Les variables du protocoles sont privées

H5. Le protocole enveloppe la section critique (Entrée; SC; Sortie) afin d'assurer son utilisation en exclusion mutuelle.

H6. Aucune hypothèse concernant la vitesse des processus à part qu'elle est différentes de 0.

Protocole 0

V_P tour = 0 ou 1;

```
P0
for(;;)
{
/* Entrée */
while ( tour != 0)  ;
SC0;
tour = 1;
}
```

//

```
P1
for(;;)
{
/* Entrée */
while ( tour != 1)  ;
SC1;
tour = 0;
}
```

Exclusion mutuelle est assurée

A rejeter H2 n'est pas respectée

Protocole 1

V_P tour;

```
P0
for(;;)
{
/* Entrée */
tour = 0;
while ( tour != 0)  ;
SC0;
tour = 1;
}
```

//

```
P1
for(;;)
{
/* Entrée */
tour = 1;
while ( tour != 1)  ;
SC1;
tour = 0;
}
```

A rejeter Exclusion mutuelle n'est pas assurée

Protocole 2

V_P D[2] = (0,0);

```
P0
for(;;)
{
/* Entrée */
D[0] = 1;
while ( D[1])  ;
SC0;
D[0] = 0;
}
```

//

```
P1
for(;;)
{
/* Entrée */
D[1] = 1;
while ( D[0])  ;
SC1;
D[1] = 0;
}
```

Exclusion mutuelle est assurée

A rejeter Livelock: attente mutuelle

Protocole 3

V_P D[2] = (0,0);

```
P0
for(;;)
{
/* Entrée */
D[0] = 1;
while ( D[1])
                    {D[0] = 0;
                     while ( D[1]);
                    D[0] = 1;
                    };

SC0;

D[0] = 0;
}
```

//

```
P1
for(;;)
{
/* Entrée */
D[1] = 1;
while ( D[0])
                    {D[1] = 0;
                     while ( D[0]);
                    D[1] = 1;
                    } ;

SC1;

D[1] = 0;
}
```

Exclusion mutuelle est assurée

A rejeter Livelock: attente mutuelle

Protocole 4

```
V_P   D[2] = (0,0);
      tour=0 ou 1;
```

```
P0
for(;;)
{
/* Entrée */
D[0] = 1;
while ( D[1])
        if (tour !=0)
                {D[0] = 0;
                 while ( D[1]);
                D[0] = 1;
                };

SC0;

D[0] = 0; tour = 1;
}
```

//

```
P1
for(;;)
{
/* Entrée */
D[1] = 1;
while ( D[0])
        if (tour !=1)
                {D[1] = 0;
                 while ( D[0]);
                D[1] = 1;
                } ;

SC1;

D[1] = 0; tour = 0;
}
```

- Exclusion mutuelle est assurée
- Pas d'attente mutuelle

A rejeter: Famine

Protocole 5
Dekker 1968

```
V_P   D[2] = (0,0);
      tour=0 ou 1;
```

```
P0
for(;;)
{
/* Entrée */
D[0] = 1;
while ( D[1])
        if (tour !=0)
                {D[0] = 0;
                 while ( tour == 1);
                D[0] = 1;
                };

SC0;

D[0] = 0; tour = 1;
}
```

//

```
P1
for(;;)
{
/* Entrée */
D[1] = 1;
while ( D[0])
        if (tour !=1)
                {D[1] = 0;
                 while ( tour == 0);
                D[1] = 1;
                } ;

SC1;

D[1] = 0; tour = 0;
}
```

- Exclusion mutuelle est assurée
- Pas d'attente mutuelle
- Pas de famine

Protocole 6
Peterso 1981

```
V_P   D[2] = (0,0);
      tour;
```

```
P0
for(;;)
{
/* Entrée */
D[0] = 1; tour = 1;
while ( (D[1]) && ( tour ==1) )   ;

SC0;

D[0] = 0;
}
```

//

```
P1
for(;;)
{
/* Entrée */
D[1] = 1; tour = 0;
while ( (D[0]) && ( tour ==0) )   ;

SC1;

D[1] = 0;
}
```

- Exclusion mutuelle est assurée
- Pas d'attente mutuelle
- Pas de famine

Problème

Boucle d'attente active → Charge des UC

Solution

Opération de blocage
À l a place
de la boucle d'attente active → performance

Semaphore de Dikstra 1968

Idée : Guichet d'un cinéma

Compteur de ticket

File d'attente

Compteur initialisé au nombre des places libres

Pour entrer on doit prendre un ticket — S'il n'y a pas de tickets attendre dans la file;
Décrémenter le compteur;

Pour sortir on rend le ticket — Incrémenter le compteur;
débloquer, éventuellement,
Si la file n'est pas vide, la tête

```
Semaphore struct {        int count;
                          fifo F; }
Init (Semaphore S,  Natural N) { S.count = N; créer(S.F);}

P(Semaphore S)
        { if (S.count == 0)
                          {inserer (num_proc_courant, S.F);
                           bloquer(proc_courant);
                          }
        S.count--;
        }

V(Semaphore S)
        { S.count ++;
        if (!vide(S.F))
                {extraire(num_proc_courant,S.F);
                debloquer(num_proc_courant);
                          }
        }
```

Init, P et V s'exécutent en exclusion mutuelle

Init joue le rôle d'un constructeur

P est libre si on est bloqué

Utilisation des sémaphores

Attendre un événement et protection des ressources

Exemple 1

```
Attendre un événement
L'événement est mémorisé

Semaphore S;
Init(S,0);

P0
{ ……../* attendre un événement */
……../* S.count > 0 */…….
                P(S);
……../* S.count ≥ 0 */…….


}
```

```
Événement
{ /* signaler  */
……./* S.count ≥ 0 */…….
V(S);
……./* S.count > 0 */…….
}
```

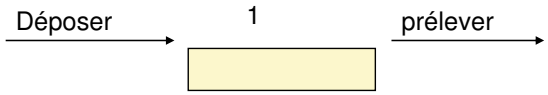
Exemple 2

```
Exclusion mutuelle
Entre N processus

Semaphore Mutex;
Init(Mutex,1);


Pi
{
P( Mutex);

SCi;

V(Mutex);
}
```

Exemple 3

Producteur
Consommateur
Buffer : un message

producteur consommateur



Ce n'est pas un problème d'exclusion mutuelle

Deux problèmes
- on ne doit pas perdre de message
- un message est consommé une et une seule fois

Problème de performance

```
message  T;        /* T est un buffer de la taille d'un message*/
Semaphore T_vide; Init(T_vide,1);
Semaphore T_plein; Init(T_plein, 0);
```

```
Producteur
{message m;
for(;;)
 {
        Construire(m);
 /*attendre, éventuellement,  que T soit vide*/
        P(T_vide);
 /* déposer m dans T */
        T = m;
 /* signaler que T contient un nouveau message*/
        V(T_plein);
 };
}
```

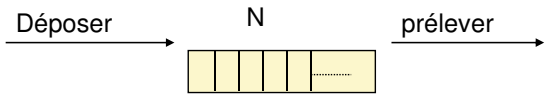
```
Consommateur
{ message m;
for(;;)
 {
 /*attendre, éventuellement,  que T soit plein*/
        P(T_plein);
 /* prélever m de T */
        m = T;
 /* signaler que T est vide*/
        V(T_vide);
        Consommer(m);
 };
}
```

Exemple 4

Producteur
Consommateur
Buffer : un message

Solution:
Augmenter la taille du Buffer

producteur consommateur



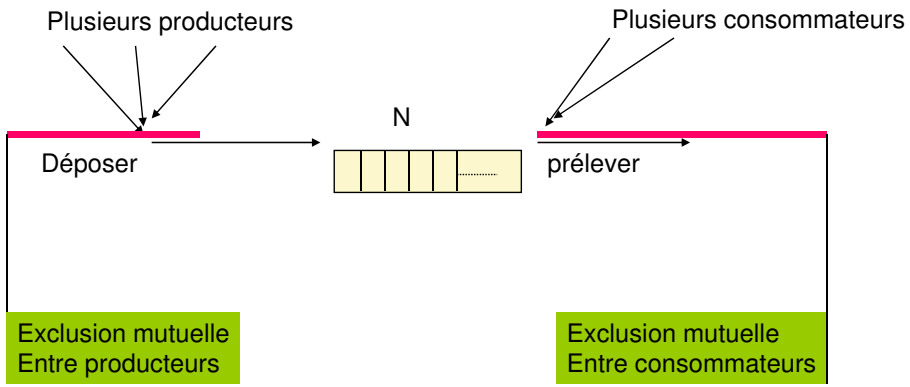

```
message  T [N];   /* T est un buffer de taille N*/
Semaphore T_vide Init(T_vide,N);
Semaphore T_plein Init (T_plein, 0);
```

```
Producteur
{message m; int I = 0;
for(;;)
 {
        Construire(m);
 /*attendre, éventuellement,  qu'une case de  T soit vide*/
        P(T_vide);
 /* déposer m dans T */
        T[I] = m; I = (I+1) % N;
 /* signaler que T contient un nouveau message*/
        V(T_plein);
 };
}
```

```
Consommateur
{ message m; int J = 0;
for(;;)
 {
 /*attendre, éventuellement,  qu'une case de  T soit pleine*/
        P(T_plein);
 /* prélever m de T */
        m = T [J]; J = (J+1)% N;
 /* signaler que T est vide*/
        V(T_vide);
        Consommer(m);
 };
}
```

Exemple 5

ProducteurS
ConsommateurS
Buffer : un message



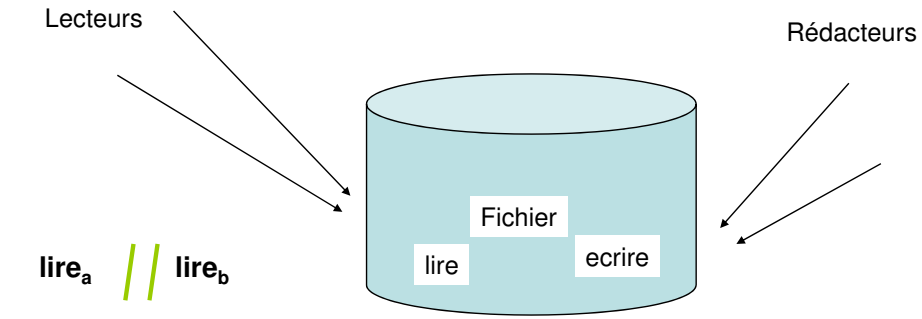

```
message  T [N];   /* T est un buffer de taille N*/
Semaphore T_vide; Init(T_vide,N);
Semaphore T_plein; Init (T_plein, 0);
Semaphore MutexP;Init(MutexP, 1);
Semaphore MutexC; Init(MutexC,1);

int I = 0; /* partagée par les producteurs
int J = 0;/* partagée par les consommateurs
```

```
Producteur_i
{message m;
for(;;)
 {
        Construire(m);
 /*attendre, éventuellement,  qu'une case de  T soit vide*/
        P(T_vide);
 /* déposer m dans T */
        P(MutexP);
        T[I] = m; I = (I+1) % N;
        V(MutexP);
 /* signaler que T contient un nouveau message*/
        V(T_plein);
 };
}
```

```
Consommateur_j
{ message m;
for(;;)
 {
 /*attendre, éventuellement,  qu'une case de  T soit pleine*/
        P(T_plein);
 /* prélever m de T */
        P(MutexC);
        m = T [J]; J = (J+1)% N;
        V(MutexC);
 /* signaler que T est vide*/
        V(T_vide);
        Consommer(m);
 };
}
```

Exemple 6

Problème des Lecteurs /Rédacteurs



Exemple 6

Comportement d'un rédacteur

Il interdit l'accès au fichier en lecture ou en écriture

On résout le problème par un sémaphore d'exclusion mutuelle

```
Semaphore w; Init(w,1);
```

```
Rédacteur
For(;;)
{ P(w); ecrire(f); V(w); }
```

Exemple 6

Comportement d'un lecteur
Il peut lire en parallèle avec d'autres lecteurs
Il doit être en exclusion mutuelle avec un rédacteur

Un lecteur effectue un P(w) pour accéder au fichier en exclusion mutuelle.

Solution 1

```
Lecteur

for(;;)
{ P(w); Lire(f); V(w); }
```

On perd le parallélisme entre lecteurs

Exemple 6

Le premier lecteur effectue un P(w) pour accéder au fichier en exclusion mutuelle.

solution2

S'il traverse cette barrière,
un autre lecteur peut le suivre.
S'il est bloqué alors un nouveau lecteur est bloqué derrière lui

Il nous faut un compteur des lecteurs en cours

```
int nlc = 0; /* auccun lecteur dans le fichier*/
```

```
If (nlc ==0) P(w);
nlc++;

Lire(F)

Nlc--;
If (nlc==0) V(w);
```

Problème pour modifier nlc entre lecteurs

Exemple 6

```
int nlc = 0; /* auccun lecteur dans le fichier*/

Semaphore w; Init(w, 1);

Semaphore MutexL; Init(MutexL, 1);
```

Lecteur

```
P(MutexL );
If (nlc ==0) P(w);
nlc++;
V(MutexL);
```

Famine Possible Des rédacteurs

```
Redacteur

for(;;)
{ P(w); ecrire(f); V(w); }
```

Lire(F)

```
P(MutexL );
nlc--;
If (nlc==0) V(w);
V(MutexL);
```

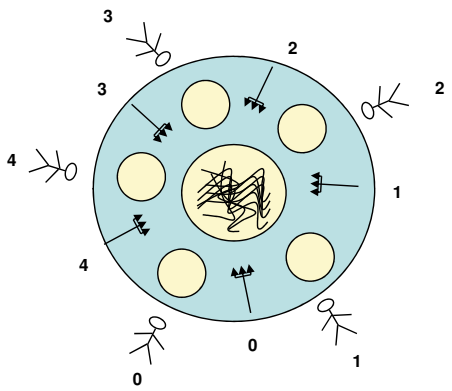
Le premier lecteur est attente

fait attendre toutes les nouvelles demandes de lectures

Le dernier lecteur lance une cascade de réveil

Exemple 7

5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette



Solution 1

```
Phi

For(;;)
{
Penser;
Prendre (Fi,Fi-1);
Manger;
Rendre (Fi,Fi-1);
}
```

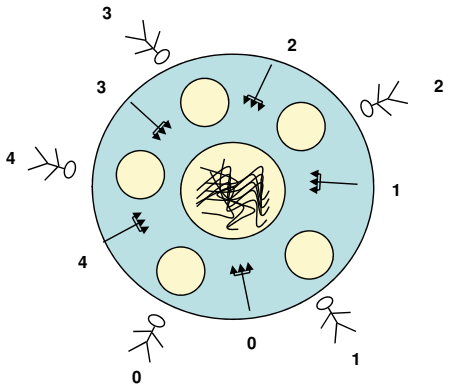
Temps fini

Atomiques
À implémenter

Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)

Exemple 7

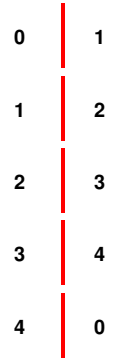
5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette



Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)

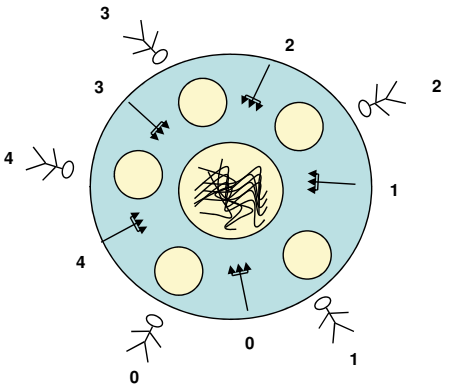
Solution 1

Exclusion mutuelle entre deux voisins



Exemple 7

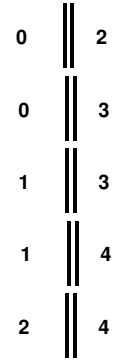
5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette



Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)

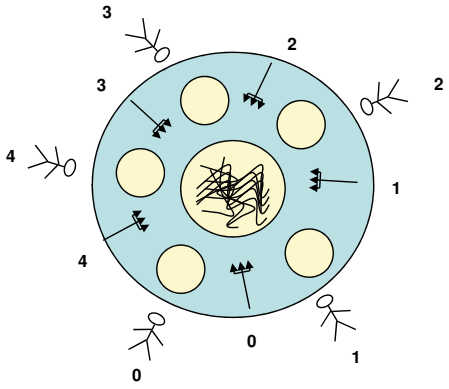
Solution 1

Parallélisme Possible entre deux Non voisins



Exemple 7

5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette



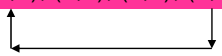
Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)

Solution 1

Exclusion mutuelle entre deux voisins

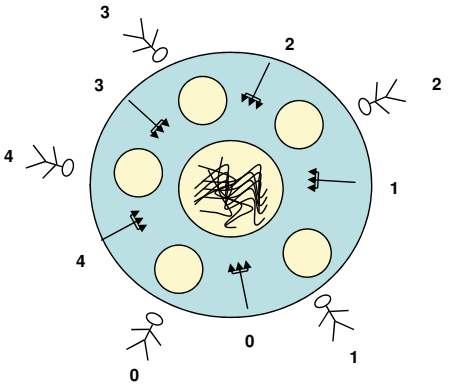
Parallélisme possible entre deux non voisins

Famine
cycle
(0,3); (0,2); (4,2); (0,2);



Exemple 7

5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette



Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)

Solution 2

Avoir 4 sur 5 à Table

Ph_i

```
For(;;)
{  se_mettre_à_table;
Penser;
Prendre (F_i,F_i-1);
Manger;
Rendre (F_i,F_i-1);
  sortir_de_la_table;
}
```

5 philosophes se sont réunis autour d'une table ronde
Pour philosopher et pour manger du spaghettis
Chaque philosophe possède une fourchette

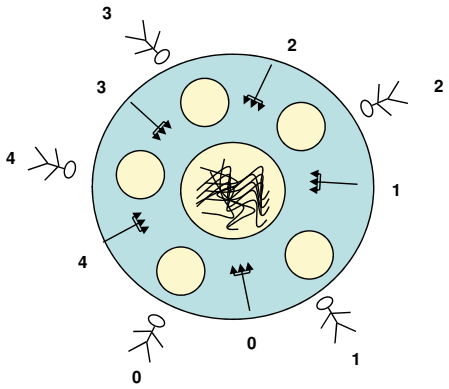
Solution 3

Prendre (Fi,Fi-1);
Est coupée en 2

Deadlock possible

Solution 4

Au moins un droitier
Et
Au moins un gaucher



Ph_d

```
for(;;)
{
Penser;
Prendre (Fi);
Prendre(Fi-1);
Manger;
Rendre (Fi);
Rendre(Fi-1);
}
```

Ph_g

```
for(;;)
{
Penser;
Prendre (Fi-1);
Prendre(Fi);
Manger;
Rendre (Fi);
Rendre(Fi-1);
}
```

Mais pour manger il faut utiliser les deux fourchettes
(sa fourchette qui est à sa droite et celle de son voisin gauche)