Chapitre 6.

6. Synchronisation avec des sémaphores

6.1. Définition

Une autre application des sémaphores consiste à synchroniser des processus qui collaborent pour réaliser une tâche suivant un ordonnancement bien défini.
Dans certaines applications informatiques, les processus doivent se synchroniser. Si la donnée à traiter n'est pas disponible, le processus chargé de la traiter doit attendre.

Le schéma général de la synchronisation par sémaphore de deux processus p1 et p2 est donné ci-dessous, le sémaphore étant initialisé à 0.

S : sémaphore initialisé à 0 ;

processus p1 ;	**processus p2 ;**
…	…
P(S) ; Puis-je	V(S) ; Vas-y
…	…

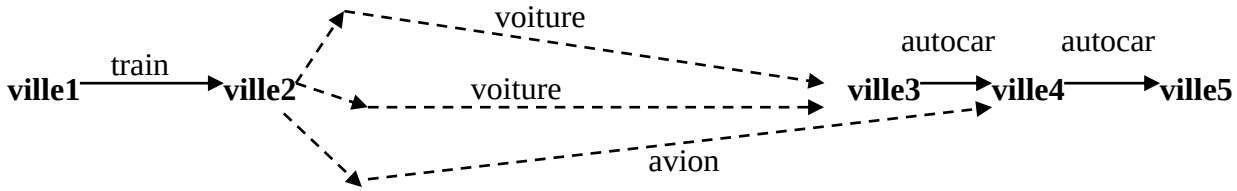
6.2. Exemple des moyens de transport

Pour aller de la ville1 à la ville5, un groupe de personnes effectuent le déplacement suivant :

ville1 – ville2 en train
puis, pour certaines personnes :
    ville2 – ville3 en voiture (il y a deux voitures) ,
    et ville3 – ville4 – ville5 en autocar
pour les autres :
    ville2 – ville4 en avion ,
    ville4 – ville5 en autocar (celui venant de la ville3)

On peut indiquer les différentes actions et synchronisations des processus : train, voiture, avion, autocar, à effectuer de façon à transporter tout le monde en utilisant les sémaphores et les opérations P et V. Il faut mettre un sémaphore là où on veut bloquer certains processus. On a donc besoin des sémaphores : ville2, ville3 et ville4. Les sémaphores sont des sémaphores de synchronisation : leur valeur initiale est 0. Chacun des processus peut démarrer dans n'importe quel ordre. Les opérations P et V doivent assurer la bonne synchronisation.

Sémaphore ville2 initialisé à 0
Sémaphore ville3 initialisé à 0
Sémaphore ville4 initialisé à 0



processus train	processus voiture	processus avion	processus autocar
Départ de ville1 … arrivée ville2 V(ville2) V(ville2) V(ville2)	P(ville2) Départ ville2 … arrivée ville3 V(ville3)	P(ville2) Décollage ville2 … arrivée ville4 V(ville4)	P(ville3) P(ville3) Départ ville3 … arrivée ville4 P(ville4) Départ ville4 … arrivée ville5

ville.c

```
#include<stdio.h>
#include<stdlib.h>
#include"tubesem.h"
Semaphore ville2, ville3, ville4;

main()
{ int i ;
 void train(int);
 void voiture(int);
 void avion(int);
 void autocar(int);
 printf("\n%-20s%-20s%-20s%-20s%-20s\n","TRAIN",
"VOITURE1","VOITURE2","AVION","AUTOCAR");

 Initsem(ville2,0);
 Initsem(ville3,0);
 Initsem(ville4,0);

 train(1);
 voiture(3);
 voiture(2);
 avion(4);
 autocar(5);
 for(i=1;i<=5;i++)wait(0); }
```

```
/* définition de train, voiture, autocar et avion */

/* train */
void train(int i)
{
if(fork()==0)
{
 message(i,"Depart de ville1");
 attente(2);
 message(i,"Arrivee a ville2");
 V(ville2);
 V(ville2);
 V(ville2);
 exit(0);
} }

/* voiture */
void voiture(int i)
{
if(fork()==0)
{
message(i,"Attente a ville2");
P(ville2);
message(i,"Depart de ville2");
attente(4);
message(i,"Arrivee ville3");
V(ville3);
exit(0);
} }

/* avion */
void avion(int i)
{
if(fork()==0)
{
message(i,"Attente ville2");
P(ville2);
message(i,"Decollage ville2");
attente(4);
message(i,"Arrivee ville4");
V(ville4);
exit(0);
} }

/* autocar */
void autocar(int i)
{
if (fork()==0)
{
```

```
message(i,"Attente ville3");
P(ville3);
P(ville3);
message(i,"Depart ville3");
attente(3);
message(i,"Arrivee ville4");
P(ville4);
message(i,"Depart ville4");
attente(2);
message(i,"Arrivee ville5");
exit(0);
} }
```

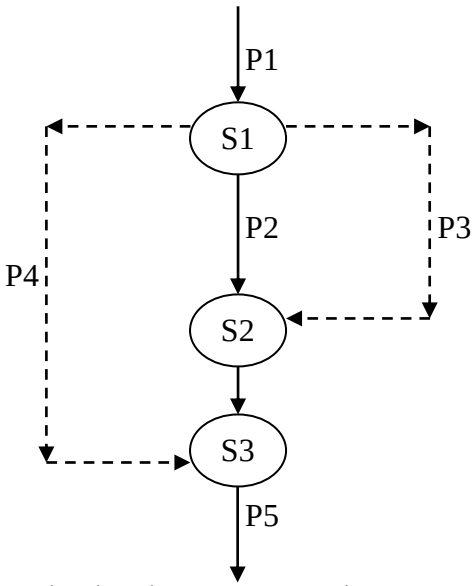
/* inclure ici les définitions des fonctions: **Initsem**, **P**, **V**, **attente** et **message** données plus haut */

Résultats :

./ville

```
TRAIN          VOITURE1       VOITURE2       AVION          AUTOCAR
Depart de ville1
                Attente a ville2
                                               Attente ville2
                                                              Attente ville3
                               Attente a ville2
Arrivee a ville2
                               Depart de ville2
                                               Decollage ville2
                Depart de ville2
                               Arrivee ville3
                                               Arrivee ville4
                Arrivee ville3
                                                              Depart ville3
                                                              Arrivee ville4
                                                              Depart ville4
                                                              Arrivee ville5
```

La synchronisation des processus peut être schématisée par le graphe d'ordonnancement suivant. Il y a 3 points de synchronisation, pour les processus P1 à P5 gérés par les trois sémaphores S1, S2 et S3.



Le graphe d'ordonnancement des processus

6.3. Exemple du producteur-consommateur

Un exemple courant de synchronisation entre processus concerne la communication de données dans une zone de mémoire partagée. Une même zone de mémoire est accessible par deux processus, l'un fournit les valeurs (processus producteur), et l'autre les utilise (processus consommateur). Les deux processus doivent se synchroniser. Le consommateur ne peut pas exploiter la valeur si celle-ci n'est pas disponible. Le producteur ne doit pas produire une nouvelle valeur si la précédente n'a pas été exploitée.

Ainsi pour une zone mémoire commune à une place :

Message	Variable commune aux 2 processus
S1	Sémaphore initialisé à 1
S2	Sémaphore initialisé à 0

processus producteur	**processus consommateur**
…	…
Puis-je produire	Puis-je consommer ?
P(S1) ;	P(S2)
Déposer les données dans message ;	Consommer les données dans message ;
Vas-y consommes	Vas-y produis
V(S2) ;	V(S1) ;
…	…

Le consommateur attend le producteur. Le producteur attend que le message soit consommé avant de produire un nouveau message. Il y a bien synchronisation des deux processus.

Pour une zone commune à N places, il suffit d'initialiser le sémaphore S1 à N (l'initialisation de S2 est maintenue à 0). La zone est une réserve de N éléments, le producteur est chargé de la remplir ; il est bloqué si la réserve est pleine. Le consommateur prend les éléments dans la réserve ; il est bloqué si la réserve est vide.