Nom : Khallouk Achraf

Travaux Pratiques N° 6

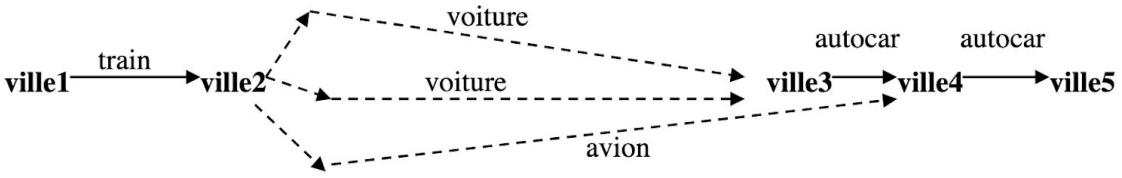
Synchronisation avec des sémaphores

Cours:

sémaphores peuvent 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.

Dans l'exemple de cours, on a utilisé trois sémaphores(vile2,ville3,ville4) qui sont des sémaphores de synchronisation, avec tous les sémaphores initialisé à 0. (Sémaphore ville2 initialisé à 0 Sémaphore ville3 initialisé à 0 Sémaphore ville4 initialisé à 0)



le programme a ete divise en deux fichier le premier "tubesem.h" qui contient les déclaration de plusieurs fonctions(P,V….etc)
et l'autre c'est ville.c où on a déclaré les fonctions voiture, train, avion qu'ils jouent le rôle des processus avec le but de incrémenter et décrémenter les valeurs des sémaphores. aussi où on definit plusieurs fonctions (message, V,P,Initsem)

ville.c (NB: le code source est sous le dossier code source avec le nom 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); }
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);
}}
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);
}}
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);
}}
void autocar(int i) {
if (fork()==0)
{
   message(i,"Attente ville3"); P(ville3);
```

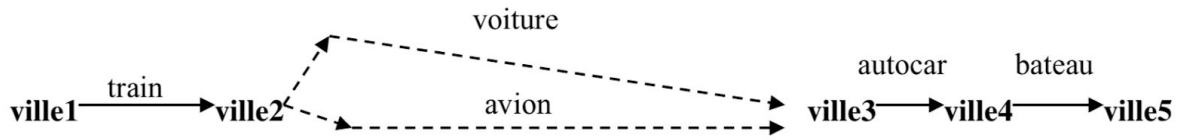
```
P(ville3); message(i,"Depart ville3"); attente(3); message(i,"Arrivee ville4");
P(ville4); message(i,"Depart ville4"); attente(2); message(i,"Arrivee ville5");
exit(0);
}}
void V (Semaphore S){
   char c='a';write(S[1],&c,1);}
   /* Attendre un nombre aleatoire de secondes entre 0 et N-1*/
   void attente(int N){
       sleep(rand() % N);
       }/* Ecrire un message s dans la i eme colonne, la premiere colonne a le
numero 1 */
       void message(int i, char* s){
           #define colonne 20
           int Nb, j;
           Nb=(i-1)*colonne;
           for(j=0; j<Nb; j++) putchar(' '); printf("%s\n" ,s);fflush(stdout);
           }
           void Initsem(Semaphore S, int N){
               int i;
               char c='a';
               pipe(S);
               for(i=1;i<=N;i++)write(S[1],&c,1);
               }
               /* P sur le semaphore S, prendre un jeton 'a' */
               void P (Semaphore S){
                   char c;read(S[0],&c,1);
                   }
```

Exécution:

```
MBP-de-DevMQK:Chapitre 6 devmqk$ ./a.out
TRAIN              VOITURE1           VOITURE2           AVION              AUTOCAR
Depart de ville1
                                      Attente a ville2
                   Attente a ville2
                                                         Attente ville2
                                                                            Attente ville3
Arrivee a ville2
                   Depart de ville2
                                      Depart de ville2
                                                         Decollage ville2
                                                         Arrivee ville4
                                      Arrivee ville3
                   Arrivee ville3
                                                                            Depart ville3
                                                                            Arrivee ville4
                                                                            Depart ville4
                                                                            Arrivee ville5
MBP-de-DevMQK:Chapitre 6 devmqk$
```

Exercice 1:

Modifier l'exemple du cours (Chapitre 6) en suivant le schéma suivant :



Processus Train	Processus Voiture	Processus avion	Processus autocar	Processus bateau
Départ de ville1 ... arrivée ville2 V(ville2) V(ville2)	P(ville2) Départ ville2 ... arrivée ville3 V(ville3)	P(ville2) Decoullage ville2 ... arrivée ville3 V(ville3)	P(ville3) P(ville3) Départ ville3 ... arrivée ville4 V(ville4)	P(ville4) Départ ville4 ... arrivée ville5

pour realise l'exercice il faut juste d'abord créer une fonction bateau(), et modifier la fonction de processus avion pour que l'avion arrive à la ville 3 ,et aussi décrémenter le nombre de l'appel de la fonction voiture et ajouter un appel bateau:

la fonction bateau:

```
void bateau(int i){
if(fork()==0)
{
message(i,"Attente ville4");P(ville4); message(i,"Depart de ville4"); attente(4);
message(i,"Arrivee ville5");
exit(0);
}}
```

ville2.c(NB: le code source est sous le dossier code source avec le nom ville2.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 bateau(int);

void autocar(int);
printf("\n%-20s%-20s%-20s%-20s%-20s\n","TRAIN",
"VOITURE","AVION","AUTOCAR","BATEAU");
Initsem(ville2,0); Initsem(ville3,0); Initsem(ville4,0);
train(1);
voiture(2);
avion(3);
autocar(4);
bateau(5);
for(i=1;i<=5;i++)wait(0); }
void train(int i) {
if(fork()==0)
{
message(i,"Depart de ville1"); attente(2);
message(i,"Arrivee a ville2");
V(ville2); V(ville2); exit(0);
}}
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);
}}
void avion(int i)
{
if(fork()==0)
{
message(i,"Attente ville2"); P(ville2); message(i,"Decollage ville2"); attente(4);
message(i,"Arrivee ville3");
exit(0);
}}
void autocar(int i) {
if (fork()==0)
{
```

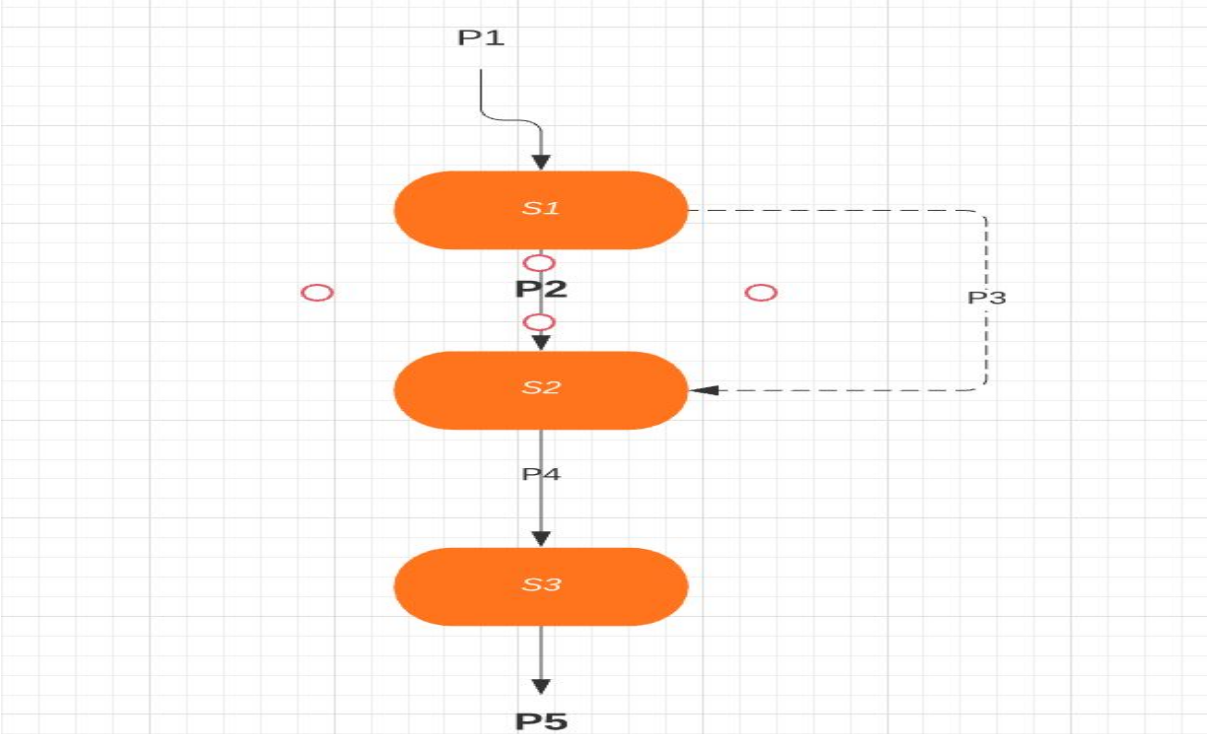
```
message(i,"Attente ville3"); P(ville3);
message(i,"Depart ville3"); attente(3);
message(i,"Arrivee ville4");
V(ville4);
exit (0);
}}
void bateau(int i){

if(fork()==0)
{
message(i,"Attente ville4");P(ville4); message(i,"Depart de ville4"); attente(4);
message(i,"Arrivee ville5");exit(0);}
}
   void V (Semaphore S){
   char c='a';write(S[1],&c,1);}
   /* Attendre un nombre aleatoire de secondes entre 0 et N-1*/
   void attente(int N){
       sleep(rand() % N);
       }/* Ecrire un message s dans la i eme colonne, la premiere colonne a le
numero 1 */
   void message(int i, char* s){
           #define colonne 20
           int Nb, j;
           Nb=(i-1)*colonne;
           for(j=0; j<Nb; j++) putchar(' '); printf("%s\n" ,s);fflush(stdout);
           }
   void Initsem(Semaphore S, int N){
               int i;
               char c='a';
               pipe(S);
               for(i=1;i<=N;i++)write(S[1],&c,1);
               }
               /* P sur le semaphore S, prendre un jeton 'a' */
   void P (Semaphore S){
                   char c;read(S[0],&c,1);
                   }
```

Exécution:

```
MBP-de-DevMQK:Chapitre 6 devmqk$ ./a.out
TRAIN               VOITURE             AVION               AUTOCAR             BATEAU
Depart de ville1
                    Attente a ville2
                                        Attente ville2
                                                            Attente ville3
                                                                                Attente ville4
Arrivee a ville2
                    Depart de ville2
                                        Decollage ville2
                                        Arrivee ville3
                    Arrivee ville3
                                                            Depart ville3
                                                            Arrivee ville4
                                                                                Depart de ville4
                                                                                Arrivee ville5
MBP-de-DevMQK:Chapitre 6 devmqk$
```

le graphe d'ordonnancement:

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.



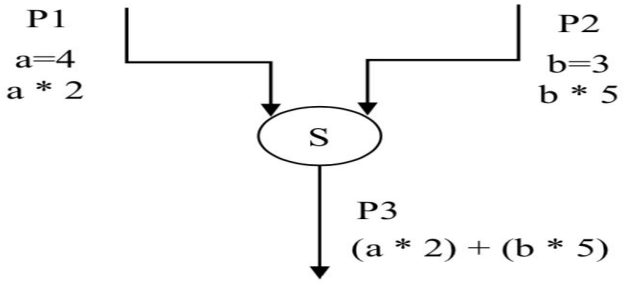
Le graphe d'ordonnancement des processus

Exercice2:

S'inspirer de l'exemple du cours (Chapitre 6), pour réaliser l'opération suivante:

(a * 2) + (b * 5) sachant que : a=4 et b=3

Cette opération nécessite la synchronisation entre trois processus (P1, P2 et P3), selon ce graphe d'ordonnancement :



pour cette exercice j'ai gardé les fonctions (message,V,P,attente,Initsem) de les exercices précédente et j'ai ajouté une fonction message appelé message2, il accepte 2 valeur int qui seront les valeurs de a et b pour afficher le message complet:

```
void message2(int i, char* s,int x,int y){
            #define colonne 20
            int Nb, j;
            Nb=(i-1)*colonne;
            for(j=0; j<Nb; j++) putchar(' ');
           printf("%s %d\n" ,s,((x * 2) + (y * 5)));fflush(stdout);
            }
```

j'ai ajouté aussi des fonctions (p1,p2,p3) qui joue le rôle des processus et qui vont incrémenter et décrémenter les valeurs de le semaphore (s) :

Processus p1	Processus p2	Processus p2
a=4 a*2 ……. V(s)	b=3 b*5 ……. V(s)	P(s),P(s) attente des valeurs ... (x * 2) + (y * 5)=

multiplication.c (NB: le code source est sous le dossier code source avec le nom multiplication.c)

```
#include<stdio.h>
#include<stdlib.h>
#include "tubesem.h"

int a=4;int b=3;
Semaphore s;
main() { int i ;
void p1(int);
void p2(int);
void p3(int);
printf("\n%-20s%-20s%-20s\n",
"a","b","Resultat");
Initsem(s,0);
p1(1);
p2(2);
p3(3);

for(i=1;i<=5;i++)wait(0); }



void p1(int i){
   if(fork()==0)
{
message(i,"a= 4");
a=a*2;attente(1);
message(i,"a*2");
V(s);
exit(0);
}}

void p2(int i)
{
   if(fork()==0)
{
message(i,"b= 3");
b*=5;attente(1);
message(i,"b*5");
V(s);
exit(0);
}}

void p3(int i){
```

```
   if(fork()==0)
{
message(i,"Attente des valeurs");
P(s);P(s);attente(3);
message2(i," (a * 2) + (b * 5) =",a,b);


exit(0);
}}



   void V (Semaphore S){
   char c='a';write(S[1],&c,1);}
   /* Attendre un nombre aleatoire de secondes entre 0 et N-1*/
   void attente(int N){
       sleep(rand() % N);
       }/* Ecrire un message s dans la i eme colonne, la premiere colonne a le
numero 1 */
   void message(int i, char* s){
           #define colonne 20
           int Nb, j;
           Nb=(i-1)*colonne;
           for(j=0; j<Nb; j++) putchar(' '); printf("%s\n" ,s);fflush(stdout);
           }

    void message2(int i, char* s,int x,int y){
           #define colonne 20
           int Nb, j;
           Nb=(i-1)*colonne;
           for(j=0; j<Nb; j++) putchar(' '); printf("%s %d\n" ,s,((x * 2) + (y *
5)));fflush(stdout);
           }
   void Initsem(Semaphore S, int N){
               int i;
               char c='a';
               pipe(S);
               for(i=1;i<=N;i++)write(S[1],&c,1);
               }
               /* P sur le semaphore S, prendre un jeton 'a' */
   void P (Semaphore S){
                   char c;read(S[0],&c,1);
```

```
}
```

Exécution:

```
MBP-de-DevMQK:Chapitre 6 devmqk$ ./a.out

a                   b                   Resultat
a= 4
a*2
                    b= 3
                    b*5
                                        Attente des valeurs
                                         (a * 2) + (b * 5) = 23
MBP-de-DevMQK:Chapitre 6 devmqk$
```