

# **Rapport Projet SE**

## **Le problème du parc d'attraction**

**Mahmoud Bacha rabah sami**

M1 RSD - GROUP : 02

## 1) Explication de rôle des sémaphores et des variables partagées utilisés :

On doit assurer le fonctionnement de N clients qui attendent de monter dans une voiture ayant P places , et que cette dernière ne puisse démarrer que si tous les places sont prisent et donc l'arrivé de P clients

- **Embarquement :**

Les clients qui arrivent doivent attendre pour voir s'ils puissent monter dans la voiture donc on utilise une sémaphore bloquante **semembarquement** , les clients se bloquent , et lorsque la voiture prend en charge la libération de P processus.

Puisque la voiture ne peut démarrer que si P clients arrivent , donc on doit implémenter un fonctionnement de barrière ou la voiture doit se bloquer jusqu'à l'arrivé de P clients , le dernier clients arrivé va libérer le processus voiture d'où l'utilisation d'une sémaphore bloquante **semtousabord** qui bloque la voiture , une variable partagée **nbr\_embarque** qui sera un compteur des clients embarqués , le dernier client va libérer la voiture qui est bloqué , le rôle du sémaphore binaire **mutex1** est de protéger l'accès concurrent a la variable partagée

### Remarque :

Lorsque un client embarque , lui aussi se bloque dans la sémaphore **semtousabord** , le dernier processus client va donc libérer les P-1 Client bloqué + le processus voiture , par la suite , il remet la variable **nbr\_embarque** a 0 pour le prochain tour.  
Car sinon , un client va être en balade juste après son embarquement !

- **Débarquement :**

De même , les clients se bloquent en attendant que la voiture termine son tour , la voiture va donc libérer les P clients qui était en balade , d'où l'utilisation d'une sémaphore bloquante **semdebarquement**

Le processus voiture doit se bloquer , en attendant que tous les P clients ont débarqué , et pour implémenter ce principe de barrière , on utilise la sémaphore bloquante **semtousdehors** , la voiture se bloque , et attend le dernier client qui débarque pour qu'elle devienne libre à faire un nouveau tour. la variable partagée **nbr\_debarque** a pour but de compter le nombre de clients débarqués, et **mutex2** est un sémaphore binaire pour protéger la variable partagée

### a) Implementation :

- Le programme Create.c contient la création et l'initialisation des sémaphores et de la mémoire partagée.  
Le programme Create doit être exécuter avant chaque réexécution pour remettre les valeurs de sémaphores et variables partagées a leur état initial.
- Le nombre de place dans la voiture est  $P = 5$  , et la durée du tours est de 10s , la voiture fait un sleep.
- La bibliothèque de la sémaphore étant déjà donnée dans le TP précédant.

#### Create :

```
#include <sys/shm.h>
#include "my_semaphore.c"

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
}sdata;

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),IPC_CREAT|IPC_EXCL|0666);

    if(shmid==-1){ //la zone existe deja !

        shmid=shmget(key,sizeof(sdata),0); //recuperer son id
        printf("Memoire partagée existe deja id : %d \n",shmid);

    }else printf("ID memoire partagée : %d\n",shmid);

    int semid = semcreate(key ,6 );
    unsigned short tabinit[6]={1,1,0,0,0,0};
    seminitall(semid,tabinit );

    printf("ID semaphore : %d\n" , semid);

    sdata *sd=NULL;

    sd=shmat(shmid,sd,0);

    // on teste si la fonction shmat a bien marché

    if (sd == NULL)
    {
```

```

    printf(" shmat didnt work ");
}else{

    sd->nbembarque=0;
    sd->nbdebarque=0;
    printf("valeurs écrites dans memoire partagé :\nnbr_embarque %d ,nbr_debarque
%d\n",sd->nbdebarque,sd->nbembarque);
}

return 0;
}

```

### Client :

```

#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
}sdata;

void embarquement(){
    printf("Client %d , je vais monter\n" , getpid());
}

void debarquement(){
    printf("Client %d , fin de balade \n" , getpid());
}

void enbalade(){
    printf("Client %d , Je suis en balade \n" , getpid());
    sleep(1);
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);
    sdata *sd=shmat(shmid,sd,0);

    int semid = semget(key , 6 , 0);
    printf("Je suis le processus Pere Client %d\nID de la memoire partage recupere :
%d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    int N ;
    printf("\n Donner le nombre de processus clients : ");scanf("%d",&N);
    printf("\n\n Mes %d fils vont commencer \n\n" , N);
}

```

```

int father_pid = getpid();

for (int i = 0; i < N; i++)
{
    if (getpid()==father_pid) fork();
}
while (1)
{
    if(getpid()!=father_pid){

        p(semid , semembarquement);
        embarquement();

        p(semid , mutex1);
        sd->nbembarque = sd->nbembarque +1;
        v(semid , mutex1);

        if (sd->nbembarque==P)
        {
            for(int i = 0 ; i < P ; i++) v(semid , semtousabord);
            sd->nbembarque = 0;
        }else{
            p(semid , semtousabord);
        }

        enbalade();

        p(semid , semdebarquement);
        debarquement();

        p(semid , mutex2);
        sd->nbdebarque = sd->nbdebarque +1 ;

        if (sd->nbdebarque == P )
        {
            v(semid , semtousdehors);
            sd->nbdebarque = 0;
        }
        v(semid , mutex2);
        sleep(1);
    }else{
        while(wait(NULL) !=-1);
    }
}
return 0;

```

**Voiture :**

```

#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

int tour = 1;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
}sdata;

void rouler(){
    printf("\nOn rouuuuuuuuule , tour %d \n" , tour);
    sleep(10);
}

void decharger(){
    printf("\nTour %d terminé , on decharge \n" ,tour );
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);
    sdata *sd=shmat(shmid,sd,0);
    int semid = semget(key , 6 , 0);

    printf("Je suis le processus Voiture %d\nID de la memoire partage recupere :
%d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    while (1)
    {
        printf("\n\n***** C'est le TOUR %d *****\n\n" , tour);

        for (int i = 0; i < P; i++)
        {
            v(semid , semembarquement);
        }
        p(semid , semtousabord);

        rouler();
        decharger();

        for (int i = 0; i < P; i++)
        {
            v(semid , semdebarquement);
        }
        p(semid , semtousdehors);
        tour++;
    }
    return 0;
}

```



Avec  $N = 7$  et  $P = 5$

5 processus peuvent embarqué : 5000 , 4998 , 5001 , 4999 , 5002

2 processus sont bloqué et qui vont être les premiers à monter dans le prochain tour : 5003 , 5004

```

Client 5002 , Je suis en balade
Client 4999 , Je suis en balade
Client 4998 , Je suis en balade
Client 5000 , fin de balade
Client 5001 , fin de balade
Client 5002 , fin de balade
Client 4998 , fin de balade
Client 4999 , fin de balade
Client 5004 , je vais monter
Client 5003 , je vais monter
Client 4999 , je vais monter
Client 5000 , je vais monter
Client 5001 , je vais monter
Client 5001 , Je suis en balade
Client 4999 , Je suis en balade
Client 5004 , Je suis en balade
Client 5000 , Je suis en balade
Client 5003 , Je suis en balade
Client 5003 , fin de balade
Client 5001 , fin de balade
Client 4999 , fin de balade
Client 5004 , fin de balade
Client 5000 , fin de balade

```

}

Tour 2

```

Je suis le processus Voiture 5011
ID de la memoire partage recupere : 45
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

On rouuuuuuuuuule , tour 1
Tour 1 terminé , on decharge

***** C'est le TOUR 2 *****

On rouuuuuuuuuule , tour 2
Tour 2 terminé , on decharge

***** C'est le TOUR 3 *****

```

Dans notre cas, les processus s'exécutent infiniment

## 2) L'utilisation de SEM UNDO :

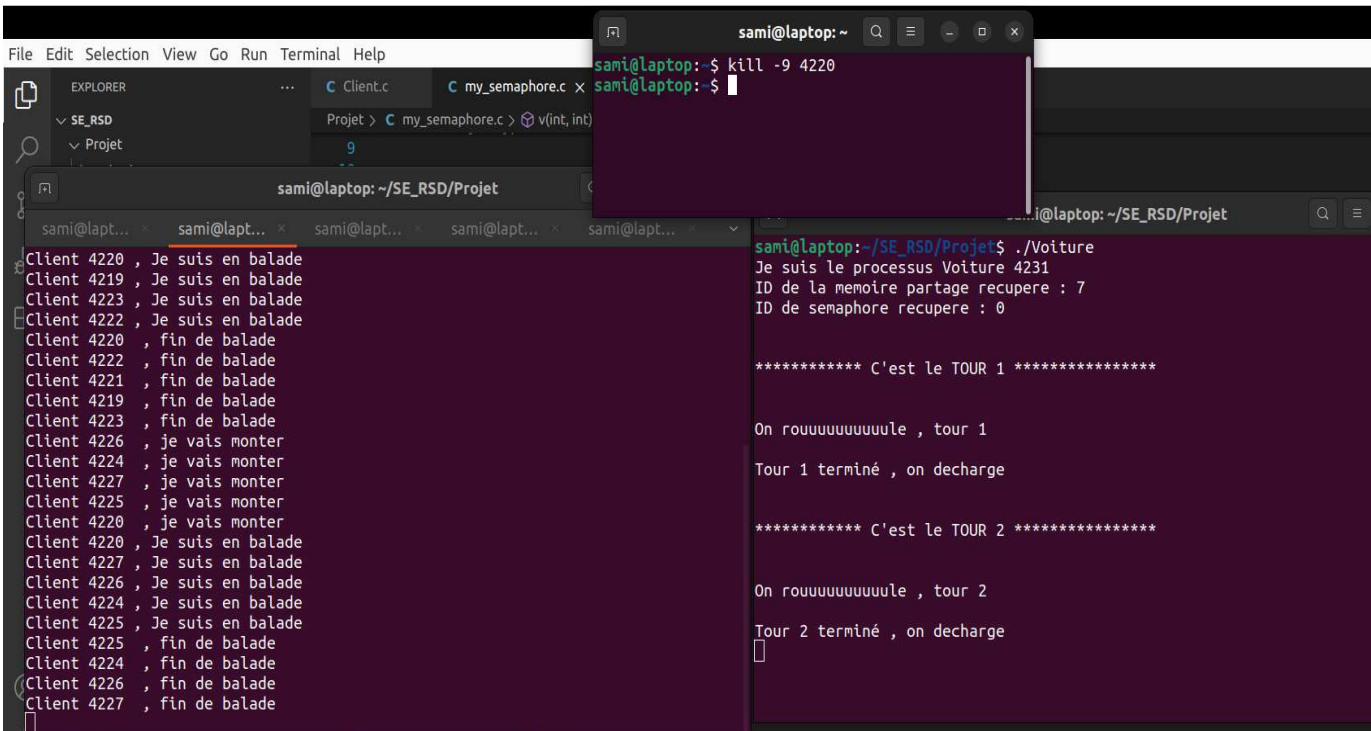
### c) Problématique :

On lance les deux processus , et on envoie un signal SIGKILL au processus client **4220** pendant qu'il était en train de faire le tour 2 .Tous les processus embarqué sont maintenant bloqué dans **semdebarquement**

Lorsque la voiture termine son tour ( après faire un sleep de 10 seconds ) , elle libérer les processus qui était en balade.

On voit bien que les clients **4224, 4225, 4226, 4227** ont été libérer par le processus voiture , mais le client **4220** est mort et donc **il ne va pas incrémenter le compteur nbr\_debarque** , Ce compteur va donc toujours rester a une valeur inferieur a P , le processus voiture reste donc bloqué dans la sémaphore **semtousdehors** , et ne peut pas passer à un autre tour et servir les prochains clients.





```

sami@laptop: ~
sami@laptop:~$ kill -9 4220
sami@laptop:~$

sami@laptop: ~/SE_RSD/Projet
Client 4220 , Je suis en balade
Client 4219 , Je suis en balade
Client 4223 , Je suis en balade
Client 4222 , Je suis en balade
Client 4220 , fin de balade
Client 4222 , fin de balade
Client 4221 , fin de balade
Client 4219 , fin de balade
Client 4223 , fin de balade
Client 4226 , je vais monter
Client 4224 , je vais monter
Client 4227 , je vais monter
Client 4225 , je vais monter
Client 4220 , je vais monter
Client 4220 , Je suis en balade
Client 4227 , Je suis en balade
Client 4226 , Je suis en balade
Client 4224 , Je suis en balade
Client 4225 , Je suis en balade
Client 4225 , fin de balade
Client 4224 , fin de balade
Client 4226 , fin de balade
Client 4227 , fin de balade

sami@laptop:~/SE_RSD/Projet$ ./Voiture
Je suis le processus Voiture 4231
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

***** C'est le TOUR 1 *****

On rouuuuuuuuuu , tour 1
Tour 1 terminé , on decharge

***** C'est le TOUR 2 *****

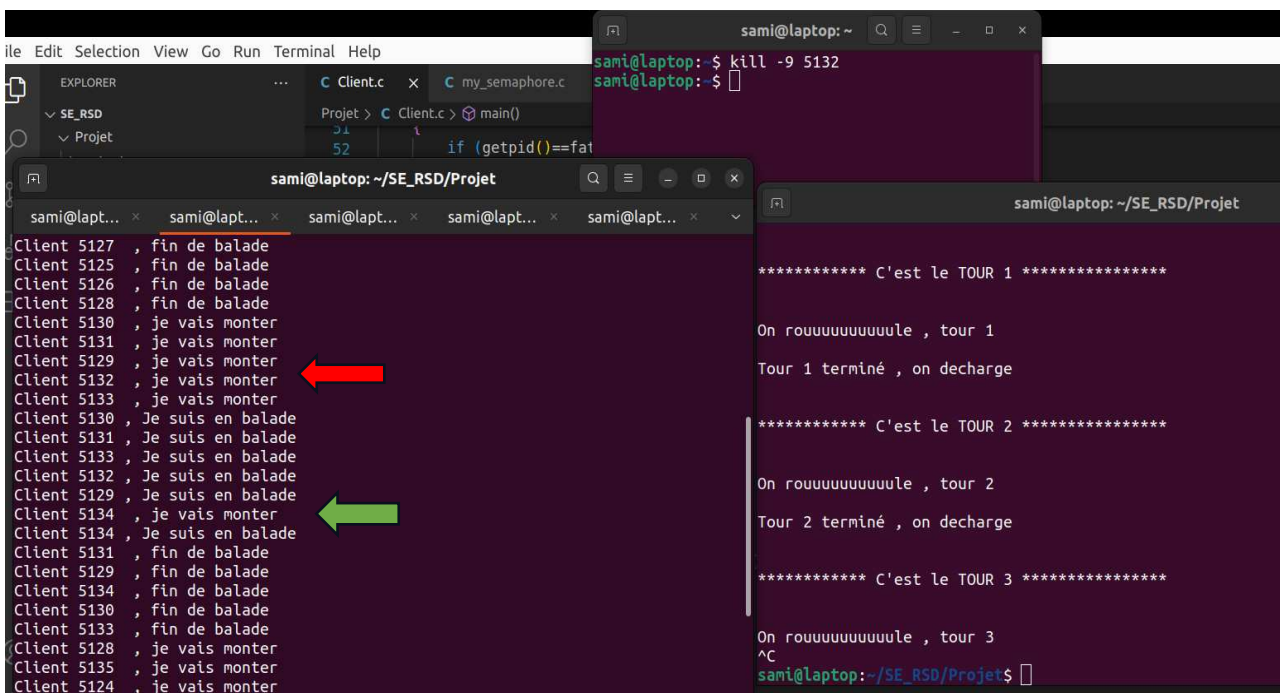
On rouuuuuuuuuu , tour 2
Tour 2 terminé , on decharge
  
```

#### d) Solution :

L'utilisation du flag **SEM\_UNDO** dans la primitive **semop** de la fonction **P()**

Lorsque ce flag est spécifié dans la fonction **semop** , le système d'exploitation peut annuler l'opération exécuter par un processus qui est accidentellement terminé.

Dans notre cas ,le OS annule tous les opérations exécuter par ce processus , l'opération de **semebarquement** va donc être annulé , le système va donc permettre au prochain client de prendre la place de cette processus mort.



```

sami@laptop: ~
sami@laptop:~$ kill -9 5132
sami@laptop:~$

sami@laptop: ~/SE_RSD/Projet
Client 5127 , fin de balade
Client 5125 , fin de balade
Client 5126 , fin de balade
Client 5128 , fin de balade
Client 5130 , je vais monter
Client 5131 , je vais monter
Client 5129 , je vais monter
Client 5132 , je vais monter
Client 5133 , je vais monter
Client 5130 , Je suis en balade
Client 5131 , Je suis en balade
Client 5133 , Je suis en balade
Client 5132 , Je suis en balade
Client 5129 , Je suis en balade
Client 5134 , je vais monter
Client 5134 , Je suis en balade
Client 5131 , fin de balade
Client 5129 , fin de balade
Client 5134 , fin de balade
Client 5130 , fin de balade
Client 5133 , fin de balade
Client 5128 , je vais monter
Client 5135 , je vais monter
Client 5124 , je vais monter

***** C'est le TOUR 1 *****

On rouuuuuuuuuu , tour 1
Tour 1 terminé , on decharge

***** C'est le TOUR 2 *****

On rouuuuuuuuuu , tour 2
Tour 2 terminé , on decharge

***** C'est le TOUR 3 *****

On rouuuuuuuuuu , tour 3
^C
sami@laptop:~/SE_RSD/Projet$
  
```

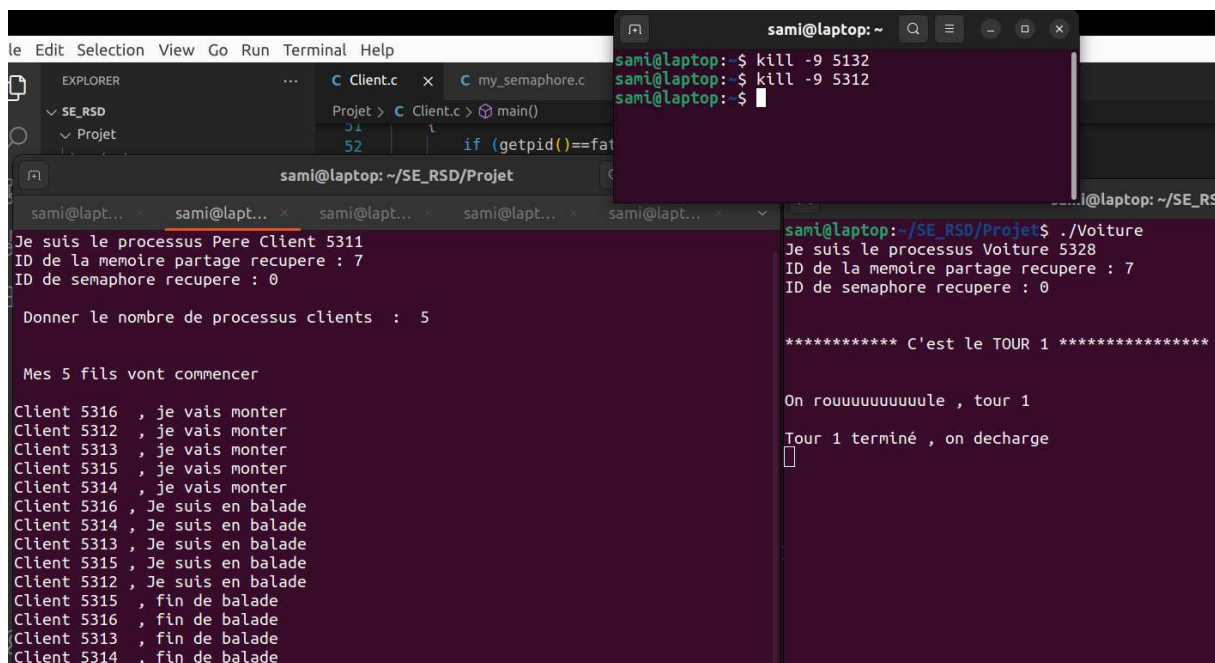
On envoie un signal au processus **5132** pendant son tour 2 , avec SEM\_UNDO , le système d'exploitation a fait lancer le premier processus qui attendait d'embarquer **5134** qui a remplacé le processus mort

Aucun processus n'est donc bloqué , la voiture peut faire ses prochains tours

### Remarque :

Un problème peut se poser si on tue un nombre de processus qui va diminuer le nombre de client jusqu'à ce que **N** devient inférieur à **P** , dans ce cas la voiture reste bloquée dans **semtousabord** , car le nombre de processus embarqué ne va jamais atteindre **P**

Par exemple  $N=5$  et on tue un processus , la voiture se bloque !



```

sami@laptop: ~$ kill -9 5132
sami@laptop: ~$ kill -9 5132
sami@laptop: ~$

sami@laptop: ~/SE_RSD/Projet
Je suis le processus Pere Client 5311
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

Donner le nombre de processus clients : 5

Mes 5 fils vont commencer
Client 5316 , je vais monter
Client 5312 , je vais monter
Client 5313 , je vais monter
Client 5315 , je vais monter
Client 5314 , je vais monter
Client 5316 , Je suis en balade
Client 5314 , Je suis en balade
Client 5313 , Je suis en balade
Client 5315 , Je suis en balade
Client 5312 , Je suis en balade
Client 5315 , fin de balade
Client 5316 , fin de balade
Client 5313 , fin de balade
Client 5314 , fin de balade

sami@laptop: ~/SE_RSD/Projet$ ./Voiture
Je suis le processus Voiture 5328
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

***** C'est le TOUR 1 *****

On rouuuuuuuuuule , tour 1
Tour 1 terminé , on decharge
  
```

## 3) Limitation de nombre de balade pour les clients :

### a) Explication de la solution :

Un petit changement a rajouter dans le processus père Client.c

On déclare une variable **client\_tour** ( non partagé ) de type integer initiales a 0.

Selon le principe d'**héritage** de la fonction `fork()` , les N processus fils créés vont tous **hérité une copie** de la variable **client\_tour** avec la même valeur d'initialisation car l'initialisation se fait avant le `fork`.

On defini dans le père une variable constante qui représente le nombre maximal de tour pour les processus fils

Ensuite , on rajoute un **test** avant de permettre au processus fils d'embarquer ( avant d'exécuter le P(embarquement) ) et une **incrémentation** juste avant de commencer la balade.

Les clients fils affiche aussi leur numéro de tour avec le message de balade , Si tous les fils du processus clients père termine leurs tour possible , le père se termine.

## b) Implémentation :

### Client.c

```
#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

#define max_tour 2

int client_tour=0;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
}sdata;

void embarquement(){
    printf("Client %d , je vais monter\n" , getpid());
}

void debarquement(){
    printf("Client %d , fin de balade \n" , getpid());
}

void enbalade(){
    printf("Client %d , Je suis en balade , c'est ma %d balade \n" , getpid() ,
client_tour);
    sleep(1);
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);
    sdata *sd=shmat(shmid,sd,0);
```

```

int semid = semget(key , 6 , 0);
printf("Je suis le processus Pere Client %d\nID de la memoire partage recupere :
%d\nID de semaphore recupere : %d \n" ,getpid() ,  shmids , semid);

int N ;
printf("\n Donner le nombre de processus clients : ");scanf("%d",&N);
printf("\n\n Mes %d fils vont commencer \n\n" , N);

int father_pid = getpid();

for (int i = 0; i < N; i++)
{
    if (getpid()==father_pid) fork();
}

while (1)
{
    if(getpid()!=father_pid ){

        if(client_tour < max_tour){
            client_tour ++;

            p(semid , semembarquement);
            embarquement();

            p(semid , mutex1);
            sd->nbembarque = sd->nbembarque +1;
            v(semid , mutex1);

            if (sd->nbembarque==P)
            {
                for(int i = 0 ; i < P ; i++) v(semid , semtousabord);
                sd->nbembarque = 0;
            }else{
                p(semid , semtousabord);
            }

            enbalade();

            p(semid , semdebarquement);
            debarquement();

            p(semid , mutex2);
            sd->nbdebarque = sd->nbdebarque +1 ;
            v(semid , mutex2);

            if (sd->nbdebarque == P )
            {
                for(int i = 0 ; i < P ; i++) v(semid , semtousdehors);
            }
        }
    }
}

```



```

        sd->nbdebarque = 0;
    }else
    {
        p(semid , semtousdehors);

    }

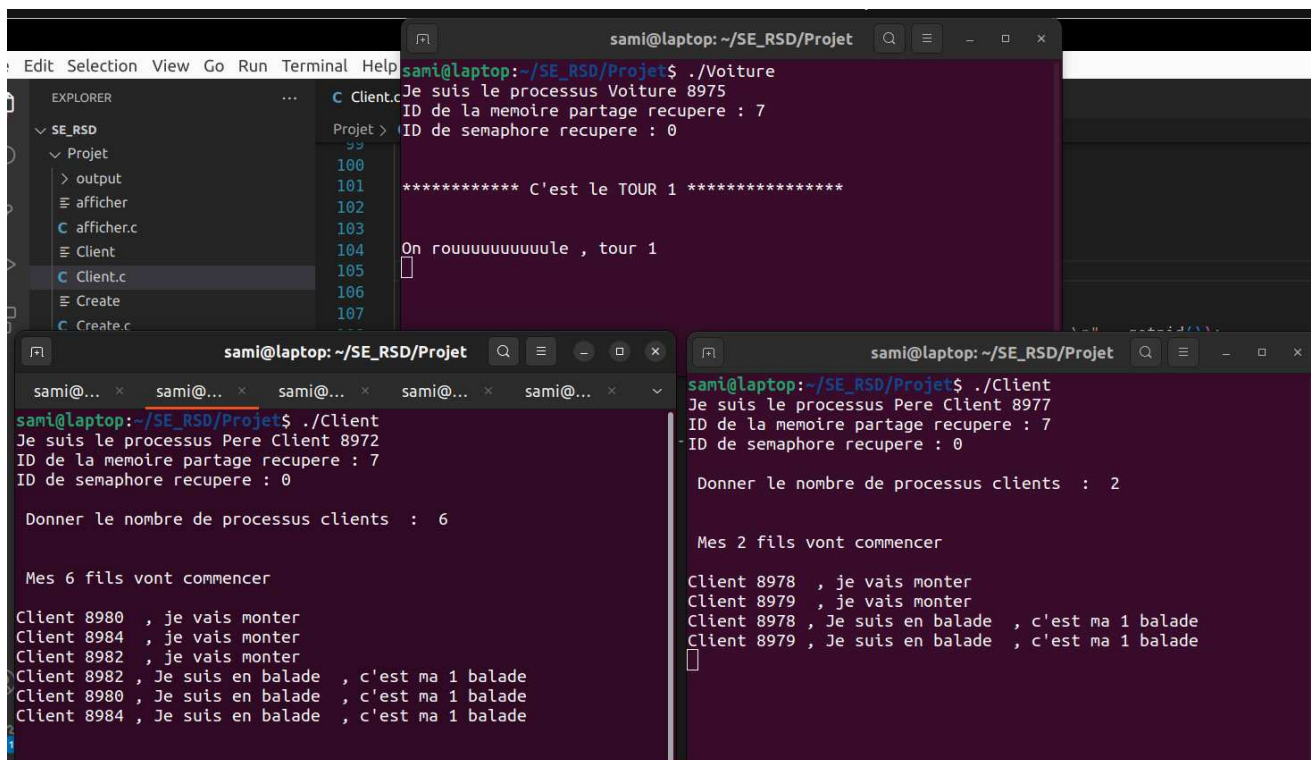
    sleep(1);
}else
{
    printf("Client %d , j'ai fait tous me tours , i die now \n" ,
getpid());
    return 0;
}

}else{
    while(wait(NULL) !=-1);
    printf("Je suis le client pere %d , tous me fils finished , i die now \n"
, getpid());

    return 0;
}
}
}
}

```

### c) Résultat d'exécution :



```

sami@laptop: ~/SE_RSD/Projet
sami@laptop:~/SE_RSD/Projet$ ./Voiture
Je suis le processus Voiture 8975
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

***** C'est le TOUR 1 *****

On rouuuuuuuuuule , tour 1
[ ]

sami@laptop:~/SE_RSD/Projet$ ./Client
Je suis le processus Pere Client 8972
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

Donner le nombre de processus clients : 6

Mes 6 fils vont commencer
Client 8980 , je vais monter
Client 8984 , je vais monter
Client 8982 , je vais monter
Client 8982 , Je suis en balade , c'est ma 1 balade
Client 8980 , Je suis en balade , c'est ma 1 balade
Client 8984 , Je suis en balade , c'est ma 1 balade

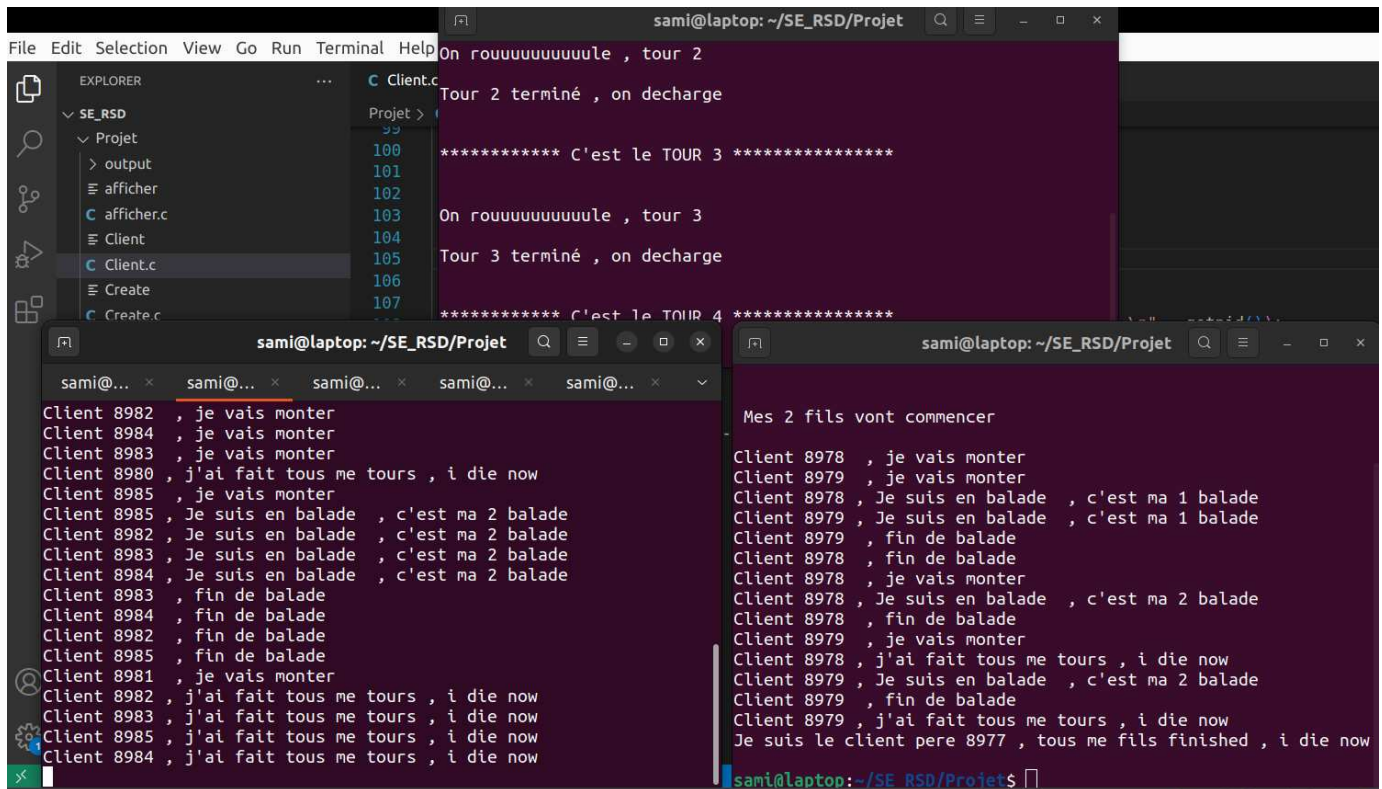
sami@laptop:~/SE_RSD/Projet$ ./Client
Je suis le processus Pere Client 8977
ID de la memoire partage recupere : 7
ID de semaphore recupere : 0

Donner le nombre de processus clients : 2

Mes 2 fils vont commencer
Client 8978 , je vais monter
Client 8979 , je vais monter
Client 8978 , Je suis en balade , c'est ma 1 balade
Client 8979 , Je suis en balade , c'est ma 1 balade
[ ]

```





```

sami@laptop: ~/SE_RSD/Projet
File Edit Selection View Go Run Terminal Help
EXPLORER
  SE_RSD
  Projet
  > output
  ≡ afficher
  C afficher.c
  Client
  Client.c
  Create
  Create.c
  Client.c
  100
  101
  102
  103
  104
  105
  106
  107
  ***** C'est le TOUR 3 *****
  On rouuuuuuuuuule , tour 3
  Tour 3 terminé , on decharge
  ***** C'est le TOUR 4 *****
  Client 8982 , je vais monter
  Client 8984 , je vais monter
  Client 8983 , je vais monter
  Client 8980 , j'ai fait tous me tours , i die now
  Client 8985 , je vais monter
  Client 8985 , Je suis en balade , c'est ma 2 balade
  Client 8982 , Je suis en balade , c'est ma 2 balade
  Client 8983 , Je suis en balade , c'est ma 2 balade
  Client 8984 , Je suis en balade , c'est ma 2 balade
  Client 8983 , fin de balade
  Client 8984 , fin de balade
  Client 8982 , fin de balade
  Client 8985 , fin de balade
  Client 8981 , je vais monter
  Client 8982 , j'ai fait tous me tours , i die now
  Client 8983 , j'ai fait tous me tours , i die now
  Client 8985 , j'ai fait tous me tours , i die now
  Client 8984 , j'ai fait tous me tours , i die now
  Mes 2 fils vont commencer
  Client 8978 , je vais monter
  Client 8979 , je vais monter
  Client 8978 , Je suis en balade , c'est ma 1 balade
  Client 8979 , Je suis en balade , c'est ma 1 balade
  Client 8979 , fin de balade
  Client 8978 , fin de balade
  Client 8978 , je vais monter
  Client 8978 , Je suis en balade , c'est ma 2 balade
  Client 8978 , fin de balade
  Client 8979 , je vais monter
  Client 8978 , j'ai fait tous me tours , i die now
  Client 8979 , Je suis en balade , c'est ma 2 balade
  Client 8979 , fin de balade
  Client 8979 , j'ai fait tous me tours , i die now
  Je suis le client pere 8977 , tous me fils finished , i die now
  sami@laptop: ~/SE_RSD/Projet$

```

On lance deux processus Clients , le premier avec 6 clients , le deuxième avec 2 clients.

**Le nombre de tour maximal est fixé à 2**

Tous les clients affichent leur numéro de balade.

Après 3 tours, les clients **8982 , 8983 , 8984 , 8985** du processus **1** et **8440 , 8441** du processus **2** ont terminé leur nombre de tour maximal. **Le processus 2 se termine** car tous ses fils sont terminés

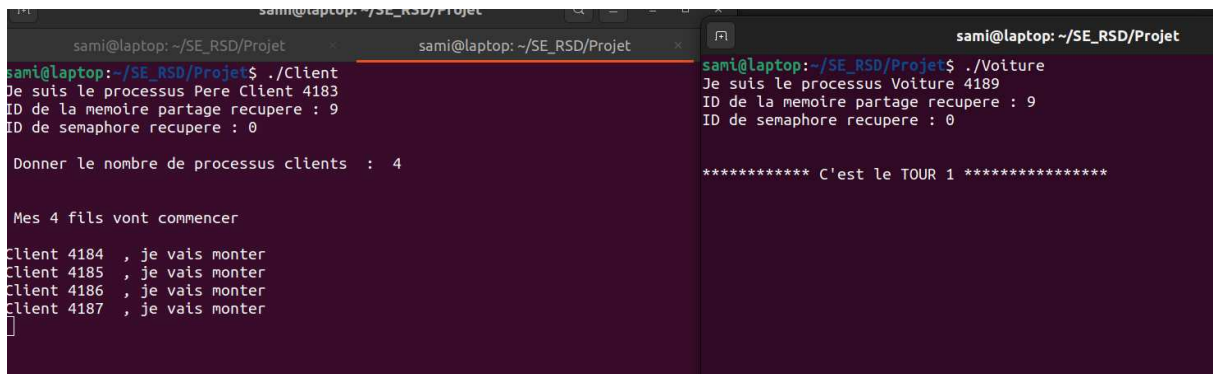
Le processus 1 reste bloqué car le fils **8981** n'a pas encore terminé sa deuxième balade ( ce processus reste bloqué car  $N < P$  )

## 4) L'utilisation de Semtimedop :

### a) Problématique :

Dans le cas où ,  $N < P$  , les  $N$  clients vont pouvoir monter , mais ils seront bloqué dans la sémaphore **semtousabord** , le processus voiture lui aussi est bloqué dans cette sémaphore.

Et donc le compteur `nbr_emabrique` ne va jamais atteindre  $P$  , Tous les processus reste donc bloqués en attendant que de nouveaux clients arrivent



```

sami@laptop: ~/SE_RSD/Projet
sami@laptop: ~/SE_RSD/Projet
sami@laptop: ~/SE_RSD/Projet$ ./Client
Je suis le processus Pere Client 4183
ID de la memoire partage recupere : 9
ID de semaphore recupere : 0

Donner le nombre de processus clients : 4

Mes 4 fils vont commencer
Client 4184 , je vais monter
Client 4185 , je vais monter
Client 4186 , je vais monter
Client 4187 , je vais monter

sami@laptop: ~/SE_RSD/Projet$ ./Voiture
Je suis le processus Voiture 4189
ID de la memoire partage recupere : 9
ID de semaphore recupere : 0

***** C'est le TOUR 1 *****
  
```

### b) Solution :

Explication de semtimedop :

Utiliser la primitive **semtimedop** au lieu de semop dans les opérations P

```
int semtimedop(int semid, struct sembuf *sops, unsigned nsops, const struct timespec *timeout);
```

- Semid est le numéro de sémaphore
- Sembuf est une structure qui définit l'opération ( incrémentation dans ce cas )
- Nsops est le numéro d'Operations ( 1 dans ce cas )
- **Timeout** est une variable de type timespec qui jouera le rôle d'un chrono, lorsque un certain délai est atteint , les processus bloqué dans cette sémaphore peuvent continuer.

La structure **timeout** contient deux variables :

- Int **tv\_sec** : le temps en seconds
- Int **tv\_nsec** : le temps en nanosecondes

La fonction **semtimedop()** se comporte comme semop() sauf que dans le cas où le processus doit dormir, la durée maximale du sommeil est limitée par la valeur spécifiée dans la structure **timespecs**. Si ce délai est atteint , le processus est réveillé et aucune opération de **\*sops** n'est réalisé et un code d'erreur **EAGAIN = 11 "Ressource temporely unavailable"** sera stocké dans la variable **errno**.

On rajoute la fonction **Ptimed()** dans le fichier header "mysemaphore.c"

```
int semtimedop(int semid, struct sembuf *sops, unsigned nsops, const struct timespec
*timeout);

int ptime (int semid , int semnum){
    struct sembuf op ;
    op.sem_num=semnum;
    op.sem_op=-1;
    op.sem_flg = 0;

    struct timespec timeout;
    timeout.tv_sec = 15; // delai d'attente fixé a 15 seconde
    timeout.tv_nsec = 0;

    int r= semtimedop(semid , &op , 1 , &timeout);
    if (r== -1) {

        if (errno == EAGAIN)
        {
            printf("Le delai de semtimedop est atteint \n");
        }else{
            perror("semtimedop");
            return 0;
        }
    }
    return r;
}
```

Scenario d'exécution :

- Si nbremarque > P :  
Une exécution normal , le dernier client va réveiller les autres clients et permettre a la voiture de lancer la balade.
- Si nbrembarque < P :  
Dans notre cas le P est fixé a 5 , la fonction **Ptime()** sera exécuté que par la voiture dans **semtousabord** , Après que le délai est atteint , deux scenarios possible :
  - nbremarqué < 3 :  
Le processus voiture va annuler le tour , libérer les clients embarqué et se remettre au début avec un goto.  
Les clients déjà embarqué vont savoir que le tour est annulé a l'aide d'une variable partagé et vont se terminer sans faire leur tour.
  - nbrembarqué >=3 ( N=4 ou N=3 ) :  
Le processus voiture lance la balade et libère les Clients pour faire leurs balade.



Pour le débarquement , la voiture attend et quand le délai est atteint elle libère les clients de **semtousdehors**

Il est important de remettre l'état des sémaphores **semdebarque semembarque semtousabord semtousdehors** dans un état correct , aussi de remettre la valeur de **errno** a sa valeur par défaut 0.

### c) Implémentation :

Voiture.c :

```
#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

int tour = 1;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
    int annule ;
}sdata;

void rouler() {
    printf("\nOn rouuuuuuuuuuule , tour %d \n" , tour);
    sleep(10);
}

void decharger() {
    printf("\nTour %d terminé , on decharge \n" ,tour );
}

int main() {

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);

    sdata *sd=shmat(shmid,sd,0);

    int semid = semget(key , 6 , 0);
    printf("Je suis le processus Voiture %d\nID de la memoire partage recupere : 
    %d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    while (1)
    {
        start :
        printf("\n\n***** C'est le TOUR %d *****\n\n" , tour);
```

```

for (int i = 0; i < P; i++)
{
    v(semid , semembarquement);
}

ptime(semid , semtousabord);

if(errno == EAGAIN){ // delai est atteint

    semctl(semid , semembarquement , SETVAL , 0); // remettre etat de
semaphore

    if(sd->nbembarque < 3){ // cas annulé

        printf("Délai dépassé et %d clients ont embarqué seuelement , le tour
est annulé\n" , sd->nbembarque);

        sd->annule = 1; // indiquer que tour annulé

        for(int i = 0 ; i < sd->nbembarque ; i++) v(semid , semtousabord);
        sd->nbembarque = 0; // liberer les clients
        sleep(2);
        sd->annule = 0;
        goto start ;

    }else{
        printf("Délai dépassé mais %d clients sont abord , on roule comme
meme \n" , sd->nbembarque);

        for(int i = 0 ; i < sd->nbembarque ; i++) v(semid , semtousabord);

        sd->nbembarque=0; // liberer les clients pour faire leur tour
    }
}

rouler();
decharger();

for (int i = 0; i < P; i++)
{
    v(semid , semdebarquement);
}

ptime(semid , semtousdehors);

if(errno == EAGAIN){ // cas de debarquement , on libere les clients

    semctl(semid , semdebarquement , SETVAL , 0);
    for(int i = 0 ; i < sd->nbdebarque;i++)v(semid , semtousdehors);
    sd->nbdebarque = 0;

}
tour++;
errno = 0 // remettre la valeur par default de errno
}
return 0;}

```

## Client.c :

```

#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

#define max_tour 2

int client_tour=0;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
    int annule;
}sdata;

void embarquement(){
    printf("Client %d , je vais monter\n" , getpid());
}

void débarquement(){
    printf("Client %d , fin de balade \n" , getpid());
}

void enbalade(){
    printf("Client %d , Je suis en balade , c'est ma %d balade \n" , getpid() ,
client_tour);
    sleep(1);
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);
    sdata *sd=shmat(shmid,sd,0);

    int semid = semget(key , 6 , 0);
    printf("Je suis le processus Pere Client %d\nID de la memoire partage recupere :
%d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    int N ;
    printf("\n Donner le nombre de processus clients : ");scanf("%d",&N);
    printf("\n\n Mes %d fils vont commencer \n\n" , N);

    int father_pid = getpid();

    for (int i = 0; i < N; i++)
    {
        if (getpid()==father_pid) fork();
    }
}

```

```

}
while (1)
{
    start :

    if(getpid() != father_pid ) {

        if(client_tour < max_tour) {

            client_tour ++;

            p(semid , semembarquement);
            embarquement();

            p(semid , mutex1);
            sd->nbembarque = sd->nbembarque +1;
            v(semid , mutex1);

            if (sd->nbembarque==P)
            {
                for(int i = 0 ; i < P ; i++) v(semid , semtousabord);
                sd->nbembarque = 0;

            }else{
                p(semid , semtousabord);
            }

            // si tour annule les clients se terminent
            if(sd->annule==1) {
                printf("Client %d , tour annulé je descend and i die \n" ,
getpid());
                return 0 ;
            }

            enbalade();

            p(semid , semdebarquement);

            débarquement();

            p(semid , mutex2);
            sd->nbdebarque = sd->nbdebarque +1 ;
            v(semid , mutex2);

            if (sd->nbdebarque == P )
            {
                for(int i = 0 ; i < P ; i++) v(semid , semtousdehors);
                sd->nbdebarque = 0;
            }else{
                p(semid , semtousdehors);
            }

            sleep(1);

```

```

    }else
    {
        printf("Client %d , j'ai fait tous me tours , i die now \n" ,
getpid());
        return 0;
    }

    }else{
        while(wait(NULL) !=-1);
        printf("Je suis le client pere %d , tous me fils finished , i die now \n"
, getpid());

        return 0;
    }
}
}
}

```

#### d) Résultat d'exécution :

- Embarqué < 3 :

On lance les deux processus avec N = 2 Au début , 2 processus embarque , ils sont bloquer dans semtousabord

La voiture aussi bloqué dans semtousabord mais avec le timer

<pre> sami@laptop:~/SE_RSD/Projet (copy)\$ ./Client Je suis le processus Pere Client 7528 ID de la memoire partage recupere : 55 ID de semaphore recupere : 1  Donner le nombre de processus clients : 2  Mes 2 fils vont commencer  Client 7530 , je vais monter Client 7529 , je vais monter </pre>	<pre> sami@laptop:~/SE_RSD/Projet (copy)\$ ./Voiture Je suis le processus Voiture 7531 ID de la memoire partage recupere : 55 ID de semaphore recupere : 1  ***** C'est le TOUR 1 ***** </pre>
---	--

Quand le délai est atteint , la voiture annule le tour et revient au début , les clients vont descendre et mourir , la voiture va chaque 15 secondes refaire le tour si le nombre de clients est insuffisant.

<pre> sami@laptop:~/SE_RSD/Projet (copy)\$ gcc Client.c -o Client sami@laptop:~/SE_RSD/Projet (copy)\$ ./Client Je suis le processus Pere Client 7528 ID de la memoire partage recupere : 55 ID de semaphore recupere : 1  Donner le nombre de processus clients : 2  Mes 2 fils vont commencer  Client 7530 , je vais monter Client 7529 , je vais monter Client 7530 , tour annulé je descend and i die Client 7529 , tour annulé je descend and i die Je suis le client pere 7528 , tous me fils finished , i die now sami@laptop:~/SE_RSD/Projet (copy)\$ </pre>	<pre> sami@laptop:~/SE_RSD/Projet (copy)\$ gcc Voiture.c -o Voiture sami@laptop:~/SE_RSD/Projet (copy)\$ ./Voiture Je suis le processus Voiture 7531 ID de la memoire partage recupere : 55 ID de semaphore recupere : 1  ***** C'est le TOUR 1 *****  semtimedop: Resource temporarily unavailable Délai dépassé et 2 clients ont embarqué seulement , le tour est annulé  ***** C'est le TOUR 1 *****  semtimedop: Resource temporarily unavailable Délai dépassé et 0 clients ont embarqué seulement , le tour est annulé  ***** C'est le TOUR 1 ***** </pre>
--	--

- Embarqué  $\geq 3$  : ( 3 ou 4 ) :

On lance un processus client et on donne  $N = 3$  , Au début les Clients sont bloqué dans **semtousabord** , de même pour la voiture.

Après 15 secondes , la voiture va démarrer

```
sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD/Projet (copy)$ gcc Voiture.c -o Voiture
sami@laptop: ~/SE_RSD/Projet (copy)$ ./Voiture
Je suis le processus Voiture 8652
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

semtimedop: Resource temporarily unavailable
Délai dépassé mais 3 clients sont aboard , on roule comme meme

On rouuuuuuuuuule , tour 1

Tour 1 terminé , on decharge
█
```

```
Donner le nombre de processus clients : 3

Mes 3 fils vont commencer

Client 8649 , je vais monter
Client 8651 , je vais monter
Client 8650 , je vais monter
Client 8649 , Je suis en balade , c'est ma 1 balade
Client 8650 , Je suis en balade , c'est ma 1 balade
Client 8651 , Je suis en balade , c'est ma 1 balade
Client 8649 , fin de balade
Client 8650 , fin de balade
Client 8651 , fin de balade
```

Les clients sont maintenant bloqué dans **semtousdehors** , après 15 secondes , ils seront débloquent pour aller au deuxième tour.

La voiture va encore attendre 15 secondes.

Pendant qu'ils sont bloqués dans **semtousabord** , on lance un autre processus client avec 3 fils.

Maintenant , le nbrembarque = 5 , le tour 2 se fait sans attente , de même pour leur débarquement .

```
sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD... x sami@laptop: ~/SE_RSD/Projet (copy)$ gcc Voiture.c -o Voiture
sami@laptop: ~/SE_RSD/Projet (copy)$ ./Voiture
Je suis le processus Voiture 8652
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

semtimedop: Resource temporarily unavailable
Délai dépassé mais 3 clients sont aboard , on roule comme meme

On rouuuuuuuuuule , tour 1

Tour 1 terminé , on decharge
semtimedop: Resource temporarily unavailable
Délai dépassé, on debarque

***** C'est le TOUR 2 *****

On rouuuuuuuuuule , tour 2
█
```

```
Client 8649 , je vais monter
Client 8651 , je vais monter
Client 8650 , je vais monter
Client 8649 , Je suis en balade , c'est ma 1 balade
Client 8650 , Je suis en balade , c'est ma 1 balade
Client 8651 , Je suis en balade , c'est ma 1 balade
Client 8649 , fin de balade
Client 8650 , fin de balade
Client 8651 , fin de balade
Client 8649 , je vais monter
Client 8651 , je vais monter
Client 8650 , je vais monter
Client 8650 , Je suis en balade , c'est ma 2 balade
Client 8649 , Je suis en balade , c'est ma 2 balade
Client 8651 , Je suis en balade , c'est ma 2 balade

Pere_Client 1

sami@laptop: ~/SE_RSD/Projet (copy)$ ./Client
Je suis le processus Pere Client 8653
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

Donner le nombre de processus clients : 3 Pere_Client 2

Mes 3 fils vont commencer

Client 8654 , je vais monter
Client 8655 , je vais monter
Client 8655 , Je suis en balade , c'est ma 1 balade
Client 8654 , Je suis en balade , c'est ma 1 balade
```



Dans le 3eme tours , Les clients **8649** , **8650** , **8651** du **Client\_1** , terminent leur 2eme tour ( maximum ) alors ils meurent.

Les clients **8654** , **8655** du **Client\_2** , font leur deuxième tour , et **8656** fait son premier

```
sami@laptop: ~/SE_RSD/Projet (copy)$ ./Voiture
Je suis le processus Voiture 8652
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

semtimeop: Resource temporarily unavailable
Délai dépassé mais 3 clients sont abord , on roule comme meme

On rouuuuuuuuuule , tour 1

Tour 1 terminé , on decharge
semtimeop: Resource temporarily unavailable
Délai dépassé, on débarque

***** C'est le TOUR 2 *****

On rouuuuuuuuuule , tour 2

Tour 2 terminé , on decharge

***** C'est le TOUR 3 *****

semtimeop: Resource temporarily unavailable
Délai dépassé mais 3 clients sont abord , on roule comme meme

On rouuuuuuuuuule , tour 3

Tour 3 terminé , on decharge
```

**8654** et **8655** terminent leur 2eme tour et **meurent** , **8656** **embarque** pour faire son deuxième tour , il est le **seul** a embarqué , le tour sera annulé donc il meurt.

```
sami@laptop: ~/SE_RSD/Projet (copy)$ ./Voiture
Je suis le processus Voiture 8652
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 4 *****

semtimeop: Resource temporarily unavailable
Délai dépassé et 1 clients ont embarqué seuelement , le tour est annulé

***** C'est le TOUR 4 *****
```

Le processus voiture reste vivant , en attendant de nouveaux clients ...

```

***** C'est le TOUR 4 *****

sentimedop: Resource temporarily unavailable
Délai dépassé et 1 clients ont embarqué seuelement , le tour est annulé

***** C'est le TOUR 4 *****

sentimedop: Resource temporarily unavailable
Délai dépassé et 0 clients ont embarqué seuelement , le tour est annulé

***** C'est le TOUR 4 *****

```

## 5) La fonction Barriere :

### a) Explication :

Le principe de cette fonction est , **N** processus se **bloquent** jusqu'à l'arrivée du **dernier processus** qui va les **libérer**.

Une variable partagé est nécessaire pour compter le nombre de processus , un mutex pour la protéger et une sémaphore bloquante pour les **N-1** processus

On pourra utiliser cette fonction pour l'embarquement dans la sémaphore semtousabord :

Les clients incrémente le nbrembarque puis , si c'est pas le dernier alors il se bloque , sinon il libère ses frères ( il libère aussi la voiture )

Pareil pour le débarquement .



## b) Implémentation :

```

void barriere_embarque (int semid , int Nombre_final , sdata sd ){

    p(semid , mutex1);
    sd.nbembarque = sd.nbembarque +1 ;
    v(semid , mutex1);

    if(sd.nbembarque == Nombre_final){
        printf("je suis le dernier %d , je les libere \n" , getpid());
        for(int i =0 ; i < Nombre_final ; i++)v(semid , semtousabord);
    }else{
        p(semid , semtousabord);
    }

}

void barriere_debarque (int semid , int Nombre_final , sdata sd ){

    p(semid , mutex2);
    sd.nbembarque = sd.nbembarque +1 ;
    v(semid , mutex2);

    if(sd.nbdebarque == Nombre_final){
        printf("je suis le dernier %d , je les libere \n" , getpid());
        for(int i =0 ; i < Nombre_final ; i++)v(semid , semtousdehors);
    }else{
        p(semid , semtousdehors);
    }

}

```

On les utilisent dans Client.c

### c) Résultat d'exécution :

```

Mes 7 fils vont commencer
Client 10499 , je vais monter
Client 10500 , je vais monter
Client 10501 , je vais monter
Client 10502 , je vais monter
Client 10505 , je vais monter
Je suis le dernier 10505 , je les libere pour embarquement
Client 10505 , Je suis en balade , c'est ma 1 balade
Client 10501 , Je suis en balade , c'est ma 1 balade
Client 10502 , Je suis en balade , c'est ma 1 balade
Client 10500 , Je suis en balade , c'est ma 1 balade
Client 10499 , Je suis en balade , c'est ma 1 balade

ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

On rouuuuuuuuuule , tour 1

Client 10502 , fin de balade
Client 10499 , fin de balade
Client 10501 , fin de balade
Client 10505 , fin de balade
Je suis le dernier 10505 , je les libere pour debarquement
Client 10504 , je vais monter
Client 10503 , je vais monter
Client 10501 , je vais monter
Client 10505 , je vais monter
Client 10500 , je vais monter
Je suis le dernier 10500 , je les libere pour embarquement
Client 10500 , Je suis en balade , c'est ma 2 balade
Client 10501 , Je suis en balade , c'est ma 2 balade
Client 10503 , Je suis en balade , c'est ma 1 balade
Client 10504 , Je suis en balade , c'est ma 1 balade
Client 10505 , Je suis en balade , c'est ma 2 balade

Je suis le processus voiture 10507
ID de la memoire partage recupere : 55
ID de semaphore recupere : 1

***** C'est le TOUR 1 *****

On rouuuuuuuuuule , tour 1

Tour 1 terminé , on decharge

***** C'est le TOUR 2 *****

On rouuuuuuuuuule , tour 2

```

### 6) Final Scripts :

#### Header file :

```

#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<sys/wait.h>
#include<errno.h>

#define mutex1 0
#define mutex2 1
#define semembarquement 2
#define semdebarquement 3
#define semtousabord 4
#define semtousdehors 5

```

```

int semtimedop(int semid, struct sembuf *sops, unsigned nsops, const struct timespec
*timeout);

int p (int semid , int semnum){
    struct sembuf op ;
    op.sem_num=semnum;
    op.sem_op=-1;
    op.sem_flg = 0;
    int r=semop(semid, &op, 1) ;
    if (r== -1) {
        perror("semop");
        return 0;
    }
    return r;
}

int ptime (int semid , int semnum){
    struct sembuf op ;
    op.sem_num=semnum;
    op.sem_op=-1;
    op.sem_flg = 0;

    struct timespec timeout;
    timeout.tv_sec = 15;
    timeout.tv_nsec = 0;

    int r= semtimedop(semid , &op , 1 , &timeout);
    if (r== -1) {
        perror("semtimedop");
        return 0;
    }
    return r;
}

void v(int semid , int semnum){
    struct sembuf op ;
    op.sem_num=semnum;
    op.sem_op+=1;
    op.sem_flg = 0;

    if (semop(semid, &op, 1) == -1) {
        perror("semop");
        exit(0);
    }
}

void z(int semid , int semnum){
    struct sembuf op ;
    op.sem_num=semnum;
    op.sem_op=0;
    op.sem_flg = 0;

```

```
if (semop(semid, &op, 1) == -1) {
    perror("semop");
    exit(0);
}

void seminit(int idsem, int numsem, int initval){
    if(semctl(idsem, numsem, SETVAL, initval) == -1) perror("semctl initialisation error");
}

void seminitall(int idsem, unsigned short initval[]){
    if(semctl(idsem, 0, SETALL, initval) == -1) perror("semctl initialisation error");
}

int semcreate(key_t key, int n){
    int semid = semget(key, n, IPC_CREAT | IPC_EXCL | 0666);
    if(semid == -1){
        perror("semget semcreate");
        semid = semget(key, n, 0);
    }
    return semid;
}

void semdestroy (int semid){
    if(semid == -1){
        printf("semaphore doesnt exist ");
    }
    else{
        semctl(semid, 0, IPC_RMID);
    }
}
```

**Create file :**

```

#include <sys/shm.h>
#include "my_semaphore.c"

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
    int annule;
} sdata;

int main() {

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),IPC_CREAT|IPC_EXCL|0666);

    if(shmid==-1){ //la zone existe deja !

        shmid=shmget(key,sizeof(sdata),0); //recuperer son id
        printf("Memoire partagée existe deja id : %d \n",shmid);

    }else printf("ID memoire partagée : %d\n",shmid);

    int semid = semcreate(key , 6 );
    unsigned short tabinit[6]={1,1,0,0,0,0};
    seminitall(semid,tabinit );

    printf("ID semaphore : %d\n" , semid);

    sdata *sd=NULL;

    sd=shmat(shmid,sd,0);

    // on teste si la fonction shmat a bien marché

    if (sd == NULL)
    {
        printf(" shmat didnt work ");
    }else{

        sd->nbembarque=0;
        sd->nbdebarque=0;
        sd->annule = 0;
        printf("valeurs écrites dans memoire partagé :\nnbr_embarque %d ,nbr_debarque %d ,annulé %d\n",sd->nbdebarque,sd->nbembarque,sd->annule);
    }

    return 0;
}

```

**Client file :**

```

#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

#define max_tour 2

int client_tour=0;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
    int annule;
}sdata;

void embarquement(){
    printf("Client %d , je vais monter\n" , getpid());
}

void débarquement(){
    printf("Client %d , fin de balade \n" , getpid());
}

void enbalade(){
    printf("Client %d , Je suis en balade , c'est ma %d balade \n" , getpid() ,
client_tour);
    sleep(1);
}

void barriere_embarque (int semid , int Nombre_final , struct data *sd ){

    p(semid , mutex1);
    sd->nbembarque = sd->nbembarque +1 ;
    v(semid , mutex1);

    if(sd->nbembarque == Nombre_final){
        printf("je suis le dernier %d , je les libere poour embarquement\n" ,
getpid());
        for(int i =0 ; i < Nombre_final ; i++)v(semid , semtousabord);
        sd->nbembarque = 0;
    }else{
        p(semid , semtousabord);
    }
}

void barriere_debarque (int semid , int Nombre_final , struct data *sd ){

    p(semid , mutex2);

```

```

sd->nbdebarque = sd->nbdebarque +1 ;
v(semid , mutex2);

if(sd->nbdebarque == Nombre_final){
    printf("je suis le dernier %d , je les libere pour débarquement\n" ,
getpid());
    for(int i =0 ; i < Nombre_final ; i++)v(semid , semtousdehors);
    sd->nbdebarque = 0 ;
}else{
    p(semid , semtousdehors);
}
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);
    sdata *sd=shmat(shmid,sd,0);

    int semid = semget(key , 6 , 0);
    printf("Je suis le processus Pere Client %d\nID de la memoire partage recupere :
%d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    int N ;
    printf("\n Donner le nombre de processus clients : ");scanf("%d",&N);
    printf("\n\n Mes %d fils vont commencer \n\n" , N);

    int father_pid = getpid();

    for (int i = 0; i < N; i++)
    {
        if (getpid()==father_pid) fork();
    }

    while (1)
    {
        if(getpid()!=father_pid ){

            if(client_tour < max_tour){

                client_tour ++;

                p(semid , semembarquement);
                embarquement();

                barriere_embarque(semid , P , sd);
            }
        }
    }
}

```

```

        if(sd->annule==1) {
            printf("Client %d , tour annulé je descend and i die \n" ,
getpid());
            return 0 ;
        }

        enbalade();

        p(semid , semdebarquement);

        débarquement();

        barriere_debarque(semid , P , sd);

        sleep(1);

    }else
    {
        printf("Client %d , j'ai fait tous me tours , i die now \n" ,
getpid());
        return 0;
    }

    }else{
        while(wait(NULL) !=-1);
        printf("Je suis le client pere %d , tous me fils finished , i die now \n"
, getpid());

        return 0;
    }
}

}

```



## Voiture file :

```
#include <sys/shm.h>
#include "my_semaphore.c"

#define P 5

int tour = 1;

typedef struct data{
    int nbembarque ;
    int nbdebarque ;
    int annule ;
}sdata;

void rouler(){
    printf("\nOn rouuuuuuuuuuule , tour %d \n" , tour);
    sleep(10);
}

void decharger(){
    printf("\nTour %d terminé , on decharge \n" ,tour );
}

int main(){

    key_t key = ftok("/home/sami/SE_RSD/semfolderprojet",10);

    int shmid=shmget(key,sizeof(sdata),0);

    sdata *sd=shmat(shmid,sd,0);

    int semid = semget(key , 6 , 0);
    printf("Je suis le processus Voiture %d\nID de la memoire partage recupere :
    %d\nID de semaphore recupere : %d \n" ,getpid() , shmid , semid);

    while (1)
    {
        start :
        printf("\n\n***** C'est le TOUR %d *****\n\n" , tour);

        for (int i = 0; i < P; i++)
        {
            v(semid , semembarquement);
        }

        ptime(semid , semtousabord);

        if(errno == EAGAIN){
```

```

    semctl(semid , semembarquement , SETVAL , 0);

    if(sd->nbembarque < 3){

        printf("Délai dépassé et %d clients ont embarqué seulement , le tour
est annulé\n" , sd->nbembarque);
        sd->annule = 1;
        for(int i = 0 ; i < sd->nbembarque ; i++) v(semid , semtousabord);
        sd->nbembarque = 0;
        sleep(2);
        sd->annule = 0;
        goto start ;

    }else{
        printf("Délai dépassé mais %d clients sont abord , on roule comme
meme \n" , sd->nbembarque);
        for(int i = 0 ; i < sd->nbembarque ; i++) v(semid , semtousabord);
        sd->nbembarque=0;

    }

}

rouler();

sd->annule =0;

decharger();

for (int i = 0; i < P; i++)
{
    v(semid , semdebarquement);
}

ptime(semid , semtousdehors);

if(errno == EAGAIN){
    printf("Délai dépassé, on débarque\n" );
    semctl(semid , semdebarquement , SETVAL , 0);
    for(int i = 0 ; i < sd->nbdebarque;i++)v(semid , semtousdehors);
    sd->nbdebarque = 0;

}

tour++;
errno = 0;

}

return 0;
}

```

ENDDD 🤖🤖🤖