

RAPPORT PROJET SYSTEME D'EXPLOITATION

Réalisé par :Ait Hammouda kenza Akli yassamine G2

*Problème de parc
d'attraction*

- **Introduction :**

- *Le problème de parc annoncé est un problème dérivant du problème classique des producteur/consommateur. Sa résolution consiste en l'utilisation des sémaphores dans le but de protéger les variables et ressources partagées, et d'éviter l'interblocage, la famine...etc.*

-

- **Objectif :**

- *Comprendre le concept des IPCs système V de Linux.*
- *Manipuler les IPCs et les implémenter pour résoudre des problèmes de synchronisation rencontrés dans les cas réels.*

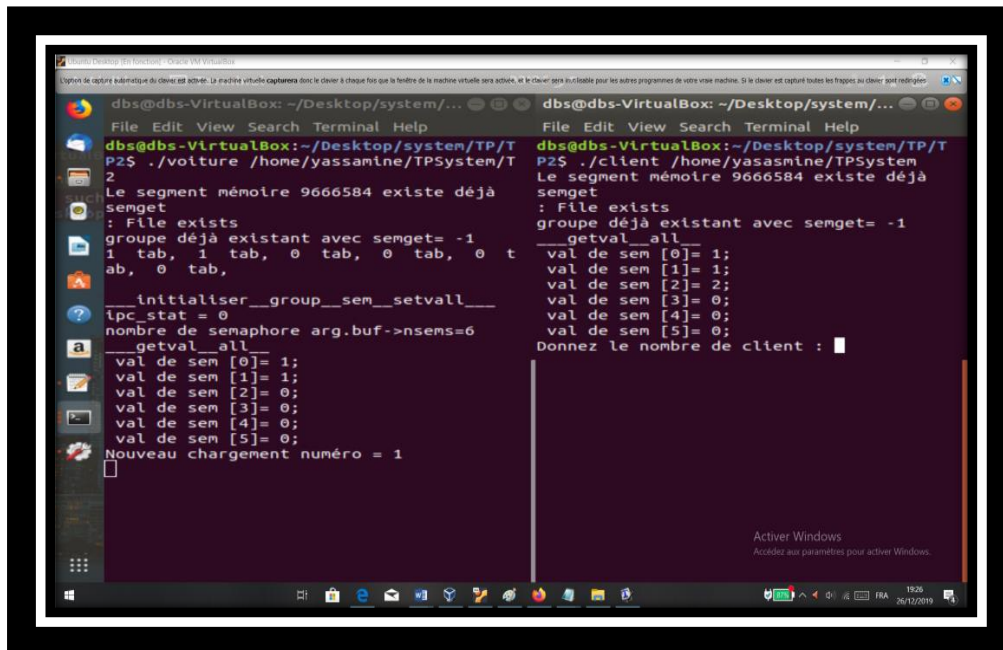
-

- **Informations sur le matériel utilisé :**

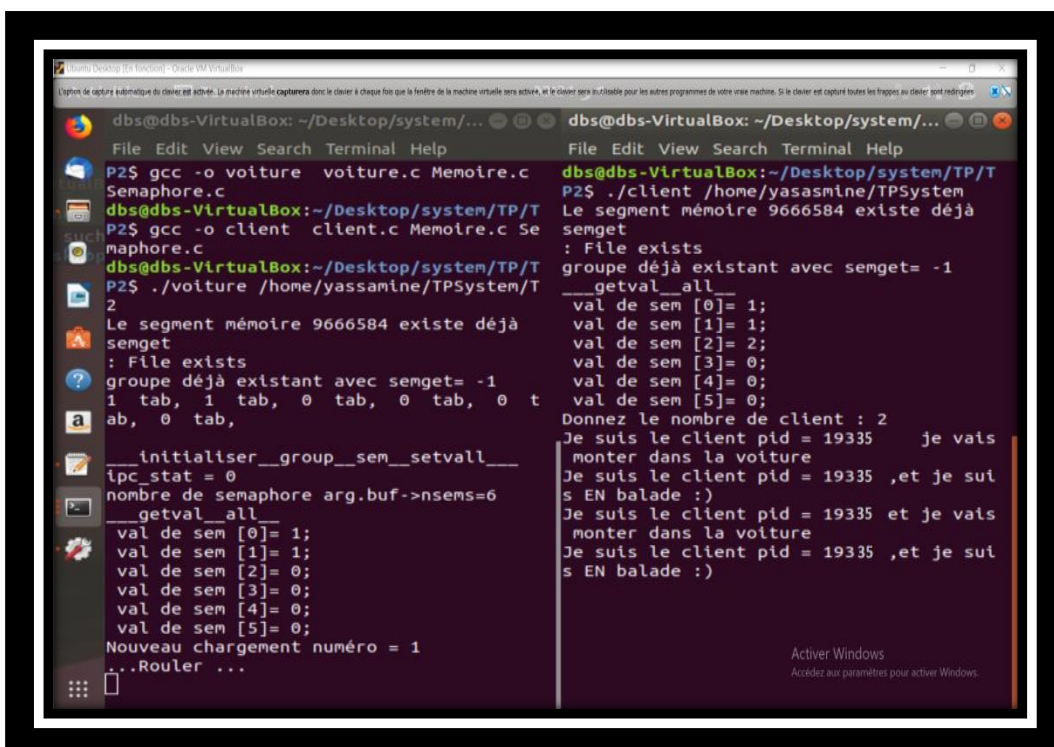
- **Ordinateur utilisé :**

- *Lenovo Z51*
- *Système d'exploitation Windows 10*
- *Processeur Intel core i5*
- *RAM : 6Go*
- ***Machine virtuelle Oracle VM VirtualBox***
- *Linux - UBUNTU (64-bit) 18.04.3 Desktop*
- *Mémoire vive : 3042 Mo*
- *USB Contrôleur : OHCI*
- *Système : Processeur ; nombre de processeur 1, ; Ressource allouées [1%-100%]*

1- Test d'exécution : $P=2, N=2$



```
dbs@dbs-VirtualBox: ~/Desktop/system/...
File Edit View Search Terminal Help
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./voiture /home/yassanine/TPSystem/TP2
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab,
__initialiser__group__sem__setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Nouveau chargement numéro = 1
Donnez le nombre de client : 1
```



```
dbs@dbs-VirtualBox: ~/Desktop/system/...
File Edit View Search Terminal Help
P2$ gcc -o voiture voiture.c Memoire.c Semaphore.c
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ gcc -o client client.c Memoire.c Semaphore.c
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./voiture /home/yassanine/TPSystem/TP2
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab,
__initialiser__group__sem__setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Nouveau chargement numéro = 1
...Rouler ...
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./client /home/yassanine/TPSystem/TP2
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 2;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 2
Je suis le client pid = 19335 je vais monter dans la voiture
Je suis le client pid = 19335 ,et je suis EN balade :)
Je suis le client pid = 19335 et je vais monter dans la voiture
Je suis le client pid = 19335 ,et je suis EN balade :)
```

```

dbs@dbs-VirtualBox: ~/Desktop/system/...
File Edit View Search Terminal Help
Nouveau chargement numéro = 3
^C
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./voiture /home/yassamine/TPSystem/T
2
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 t
ab, 0 tab,
__initialiser_group_sem_setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
__getval_all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Nouveau chargement numéro = 1
...Rouler ...
La voiture est pret a etre décharger
Nouveau chargement numéro = 2

dbs@dbs-VirtualBox:~/Desktop/system/...
File Edit View Search Terminal Help
NI de balade je doit descendre :(
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./client /home/yassamine/TPSystem
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval_all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 2;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 2
Je suis le client pid = 19335 et je vais
monter dans la voiture
Je suis le client pid = 19335 ,et je sui
s EN balade :)
Je suis le client pid = 19334 et je vais
monter dans la voiture
Je suis le client pid = 19334 ,et je sui
s EN balade :)
Je suis le client pid = 19335 et j al FI
NI de balade je doit descendre :(
Je suis le client pid = 19334 ,et j al FI
NI de balade je doit descendre :(

```

2- Réponse à la question : Qu'est ce qui se passera si on envoie un signal kill a un des processus client pendant que la voiture est en tournée ???

Le processus voiture et les processus clients seront bloqués.

Explication :

Alors le processus voiture sera bloqué en attente du dernier client qui va le débloquent à la fin de sa tournée (donc il sera bloqué dans la file d'attente de sémaphore *semTousDehors*) et les autres processus clients seront bloqués dans la file d'attente de sémaphore *semEmbarquement* en attente d'embarquer dans la voiture.

Capture d'écran pour une meilleure explication :

```

void P_TIMED(int semid, unsigned short semnum)
{
    struct sembuf ops[1];
    ops[0].sem_op=-1;
    ops[0].sem_num=semnum;
    ops[0].sem_flg=SEM_UNDO;
    struct timespec timeout;
    timeout.tv_sec=10; //dans le cas ou
    //le processus dormirait la durée de sommeil
    //est limitée par le temps écoulé spécifié par
    //la structure passer dans argument timeout
    //(cet intervalle de sommeil sera arrondi à
    //la granularité, et les retard de planification
    //du noyau signifient que
    //l'intervalle peut etre dépassé par une petite
    //quantité) si le délai spécifié a été atteint
    // semtimedop() echoue avec errno définit sur
    // EAGAIN (et aucune des opérations dans sops n'est
    //effectuée). si l'argument timeout esst NULL,
    //setimedop() se comporte exactement comme semop()
    int r=semtimedop(semid, ops, 1, &timeout);
    if(r== -1){perror("semop error p timed: ");}
    else{
    }
}
//

```

Test :

Pour les valeurs :

$P=2$, $N=2$

```

dbs@dbs-VirtualBox: ~/Desktop/system/...
File Edit View Search Terminal Help
dbs@dbs-VirtualBox: ~/Desktop/system/TP/TP2
~/Desktop/system/TP/TP2$ kill -9 19502
~/Desktop/system/TP/TP2$
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 t
ab, 0 tab,
__initialiser_group_sem_setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
__getval_all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Nouveau chargement numéro = 1
...Rouler ...
La voiture est pret a etre décharger
NI de balade je doit descendre :(
Je suis le client pid = 19334 et j ai FI
NI de balade je doit descendre :(
dbs@dbs-VirtualBox:~/Desktop/system/TP/T
P2$ ./client /home/yasasmine/TPSystem
Le segment mémoire 9666584 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval_all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 2;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 2
Je suis le client pid = 19502 et je vais
monter dans la voiture
Je suis le client pid = 19502 ,et je sui
s EN balade :)
Je suis le client pid = 19501 et je vais
monter dans la voiture
Je suis le client pid = 19501 ,et je sui
s EN balade :)
Je suis le client pid = 19501 et j ai FI
NI de balade je doit descendre :(

```


Remarque : Blocage juste après avoir débarqué le client 19501.

- Le principe de flag SEM_UNDO :

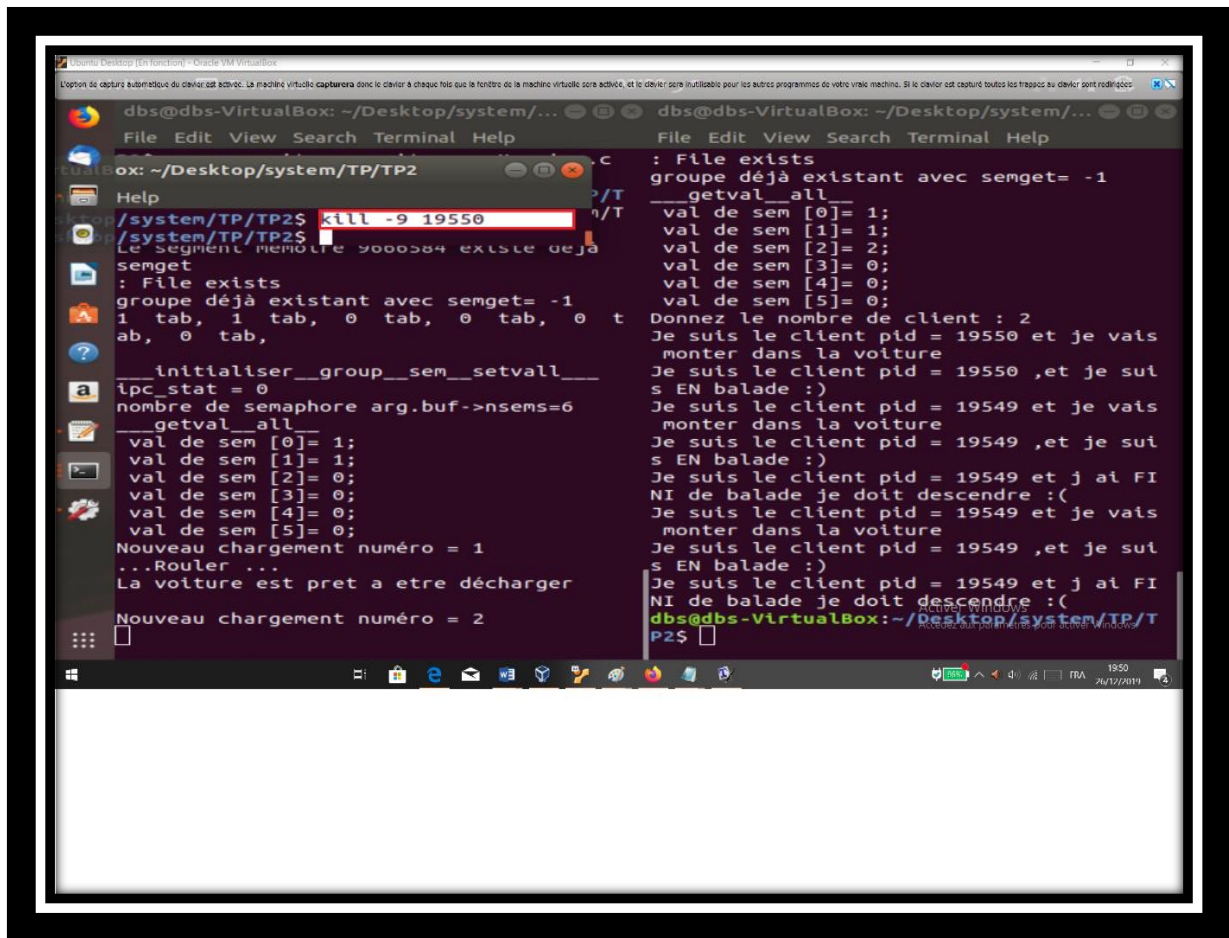
Chaque processus dispose d'une liste de valeurs *semadj* (une valeur pour chaque sémaphore ou SEM_UNDO a été utilisé).

SEM_UNDO indique que l'opération réalisée sera annulée lorsque le processus se terminera. Ceci permet d'éviter de bloquer définitivement une ressource si un processus se termine anormalement après avoir verrouillé le sémaphore.

Cette capture d'écran illustre le test après avoir modifier et rajouter le UNDO.

*Pour les valeurs :

$P=2, N=10$



3-Limitation de nombre de tournée : rajouter la variable partagée `nbr_tour` :

Explication :

Le processus voiture initialise la variable `nbr_tour`. Le processus client récupère la valeur de cet dernier et la stocke dans une variable local sur la quelle fait le test.

Incrémenter la valeur de la variable

Si (la variable égale au nombre du tourne) alors `exit(0)`

Sinon monter dans la voiture.

Capture d'écran pour une meilleure explication :

```

        nb++;
        if(nb>sd->nbtour){exit(0);}
    }
    }else{
        fils[i]=pid;
    }
}

//Attente de pere de ses fils
for(int i = 0; i < nt; i++){
    waitpid(fils[i],&status,0);
}

```

**Pour les valeurs :*

$P=2$, $N=2$, $NbTour=2$

```

dbs@dbs-VirtualBox: ~/Desktop/system/TP/T
File Edit View Connection failed Activation of network connection failed Help
senget : File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab, 0 tab,
__initialiser_group__sem__setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsens=6
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
donner le nombre de tour du client
2
...chargement... numéro = 1
...Rouler ...
...décharger...
...chargement... numéro = 2
...Rouler ...
...décharger...
...chargement... numéro = 3
dbs@dbs-VirtualBox: ~/Desktop/system/TP/T
P2$ ./client3 /home/yasasnine/TPSystem
Le segment mémoire 9666584 existe déjà
senget : File exists
groupe déjà existant avec semget= -1
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 2
EMBARQUEMENT client pid = 21805
BALADE client pid = 21805;)
EMBARQUEMENT client pid = 21804
BALADE client pid = 21804;)
FIN client pid = 21805 :(
FIN client pid = 21804 :(
EMBARQUEMENT client pid = 21805
BALADE client pid = 21805;)
EMBARQUEMENT client pid = 21804
BALADE client pid = 21804;)
FIN client pid = 21805 :(
FIN client pid = 21804 :(
dbs@dbs-VirtualBox: ~/Desktop/system/TP/T
P2$

```


4- Que se passerait-il s'il n'y a pas p client pour monter dans la voiture?

Le processus voiture sera bloqué en attente de dernier client pour qu'il puisse démarrer (le processus voiture sera inséré dans la file de sémaphore *semTousAbord*.)

Les processus clients embarqué dans la voiture seront en attente de débarquement.

```
dbb@dbb-VirtualBox:~/yassamine/TPSystem/TP2/qs2et1et3$ ./client /
Le segment mémoire 5537812 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 2;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 1
EMBARQUEMENT client pid = 7899
BALADE client pid = 7899:)
```

□

```
dbb@dbb-VirtualBox: ~/yassamine/TPSystem/TP2/qs2et1et3
File Edit View Search Terminal Help
Semaphore.c Memoire.c
dbb@dbb-VirtualBox:~/yassamine/TPSystem/TP2/qs2et1et3$ ./voiture /
Le segment mémoire 5537812 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
1 tab, 1 tab, 0 tab, 0 tab, 0 tab, 0 tab,
__initialiser__group__sem__setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
VOITURE...chargement... numéro = 1
```

La raison du problème pose :

Le processus voiture ne peut démarrer si et seulement si toutes les places sont pleines c-à-d il sera débloqué par le dernier processus client qui monte dans la voiture en exécutant $V(semTousAbord)$ si $nbEmbarques == p$, cette condition ne sera jamais vraie ($n < p$) donc le processus voiture reste toujours en attente (bloqué) du dernier client.

De ce fait le processus voiture ne peut pas débloquer les processus clients qui font la queue pour descendre.

Solution avec la fonction `semtimedop` :

Explication :

```
void P_TIMED(int semid, unsigned short semnum)
{
    struct sembuf ops[1];
    ops[0].sem_op=-1;
    ops[0].sem_num=semnum;
    ops[0].sem_flg=SEM_UNDO;
    struct timespec timeout;
    timeout.tv_sec=10; //dans le cas ou
    //le processus dormirait la durée de sommeil est
    //limitée par le temps écoulé spécifié par la structure
    //passer dans argument timeout (cet intervalle de sommeil sera arrondi à la granularité, et les retard de planification du noyau
    signifient que
    //l'intervalle peut être dépassé par une petite quantité) si le
    //délai spécifié a été atteint, semtimedop()
    //échoue avec errno défini sur EAGAIN (et aucune des opérations dans sops n'est
    //effectuée). si l'argument timeout est NULL, setimedop()
    //se comporte exactement comme semop()
    int r=semtimedop(semid, ops, 1, &timeout);
    if(r==-1){perror("semop error ptined: ");}
    else{printf("After Z %d ",r);}
}
```

5- Implémentation de la question 4:

Après l'exécution de `Ptimed(semTousAbord)` le processus voiture va faire une vérification :

Sí (`nbEmbarques`>`p / 3`) alors :

Réinitialisation de sémaphore `semEmbarquement` à 0 et libération des processus client qui font la chaîne pour descendre, après déchargement il réinitialise les valeurs de `nbEmbarques` et `nbDebarques` à zéro.

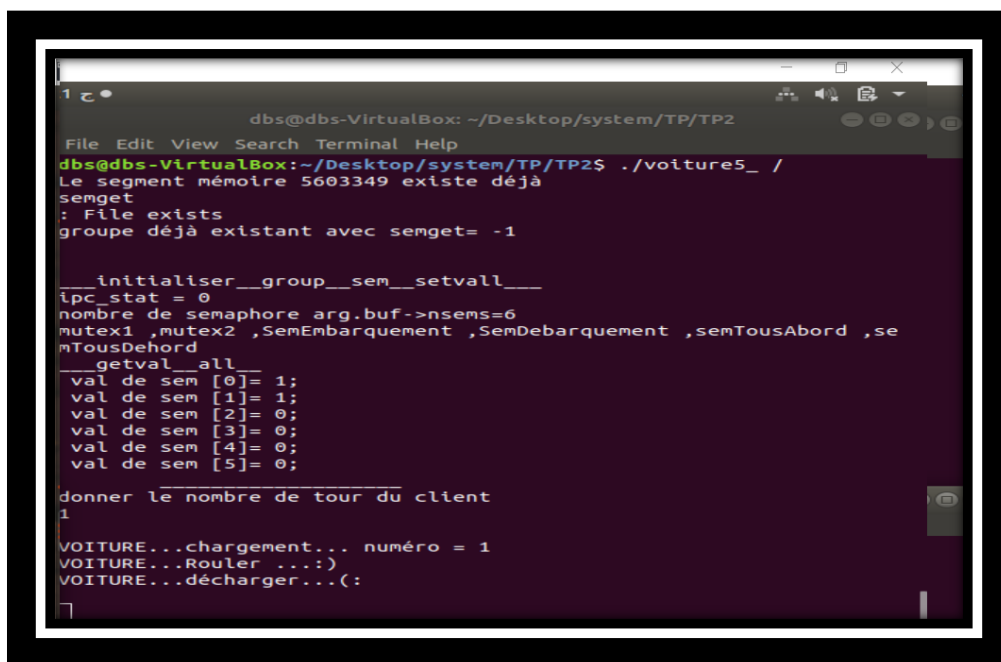
Sinon :

Affichage d'un message d'annulation et réinitialisation des sémaphores au début.

Capture d'écran qui illustre le résultat :

`P=2, N=2, NbTour=1`

`Voiture :`



```
dbs@dbs-VirtualBox: ~/Desktop/system/TP/TP2
File Edit View Search Terminal Help
dbs@dbs-VirtualBox:~/Desktop/system/TP/TP2$ ./voiture5_ /
Le segment mémoire 5603349 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1

__initialiser__group__sem__setvall__
ipc_stat = 0
nombre de semaphore arg.buf->nsems=6
mutex1 ,mutex2 ,SemEmbarquement ,SemDebarquement ,semTousAbord ,se
mTousDebord
__getval__all__
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 0;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;

donner le nombre de tour du client
1
VOITURE...chargement... numéro = 1
VOITURE...Rouler ...:)
VOITURE...décharger...(:
```

`Clients :`

```
Ubuntu Desktop [En fonction] - Oracle VM VirtualBox
Activités Terminal
dbs@dbs-VirtualBox: ~/Desktop/system/TP/TP2
File Edit View Search Terminal Help
dbs@dbs-VirtualBox:~/Desktop/system/TP/TP2$ ./client5_ /
Le segment mémoire 5603349 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval__all
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 2;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 1
EMBARQUEMENT client pid = 3856
BALADE client pid = 3856 :)
FIN client pid = 3856 :(
dbs@dbs-VirtualBox:~/Desktop/system/TP/TP2$
```

```
val de sem [5]= 0;
Donnez le nombre de client : 1
EMBARQUEMENT client pid = 3617
BALADE client pid = 3617 :)
FIN client pid = 3617 :(
^C
dbs@dbs-VirtualBox:~/Desktop/system/TP/TP2$ ./client5_ /
Le segment mémoire 5603349 existe déjà
semget
: File exists
groupe déjà existant avec semget= -1
__getval__all
val de sem [0]= 1;
val de sem [1]= 1;
val de sem [2]= 1;
val de sem [3]= 0;
val de sem [4]= 0;
val de sem [5]= 0;
Donnez le nombre de client : 1
EMBARQUEMENT client pid = 3859
BALADE client pid = 3859 :)
FIN client pid = 3859 :(
```

Remarque : Ici Le Nombre de client est égale à 1 dans chaque processus client, grace à la fonction SEMTIMEDOP les deux peuvent entrer et la voiture peut rouler.

Capture d'écran qui illustre le résultat encore plus :

$P=7$, $N=5$, $NbTour=2$

```
Donnez le nombre de client : 5
EMBARQUEMENT client pid = 6969
BALADE client pid = 6969 :)
EMBARQUEMENT client pid = 6968
BALADE client pid = 6968 :)
EMBARQUEMENT client pid = 6970
BALADE client pid = 6970 :)
EMBARQUEMENT client pid = 6967
BALADE client pid = 6967 :)
EMBARQUEMENT client pid = 6966
BALADE client pid = 6966 :)
FIN client pid = 6968 :(
FIN client pid = 6970 :(
FIN client pid = 6969 :(
FIN client pid = 6967 :(
FIN client pid = 6966 :(

EMBARQUEMENT client pid = 6970
BALADE client pid = 6970 :)
EMBARQUEMENT client pid = 6968
BALADE client pid = 6968 :)
EMBARQUEMENT client pid = 6969
BALADE client pid = 6969 :)
EMBARQUEMENT client pid = 6967
BALADE client pid = 6967 :)
EMBARQUEMENT client pid = 6966
BALADE client pid = 6966 :)
FIN client pid = 6970 :(
FIN client pid = 6967 :(
FIN client pid = 6968 :(
FIN client pid = 6969 :(
FIN client pid = 6966 :(

dbs@dbs-VirtualBox:~/Desktop/svs

...ANNULER TOURNEE...
VOITURE...décharger...(:

VOITURE...chargement... numéro = 13
VOITURE...Rouler ...:))
VOITURE...décharger...(:

semop error ptimed: : Invalid argument
VOITURE...chargement... numéro = 14

...ANNULER TOURNEE...
VOITURE...décharger...(:

VOITURE...chargement... numéro = 15

...ANNULER TOURNEE...
VOITURE...décharger...(:

VOITURE...chargement... numéro = 16
VOITURE...Rouler ...:))
VOITURE...décharger...(:

semop error ptimed: : Invalid argument
VOITURE...chargement... numéro = 17

...ANNULER TOURNEE...
VOITURE...décharger...(:


VOITURE...chargement... numéro = 18

...ANNULER TOURNEE...
VOITURE...décharger...(:

VOITURE...chargement... numéro = 19
```

Conclusion :

A chaque problème rencontré nous avons rajouté une solution adéquate .



Pour l'interblocage : soit un blocage d'un des clients en plein exécution spot le nombre de client est inférieur à celui de nombre de place. La solution à proposé pour le 1er est l'introduction du flag SEM_UNDO. Pour le deuxième c'est l'ajout de SEMTIMEDOP pour donnez encore plus de temps avant de se bloqué.

Pour la suite nous avons rajouter le nombre de tour pour des raisons d'amélioration.

Une amélioration pour éviter le blocage de voiture en cas de nombre de client est inférieur au nombre de place, donc pas la peine d'attendre le remplissage complet des place il suffira de vérifier que le nombre de client ou d'embarquement est supérieur au tier des nombres de place.