

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI  
BOUMEDIENE



SYSTÈME D'EXPLOITATION

---

**Rapport de Projet**  
**Le problème du Parc d'attraction.**

---

*Monôme*  
HOUACINE NAILA AZIZA  
Matricule : 201400007594  
M1 SII groupe :4

*Mme*  
MEHDI MALIKA

28 décembre 2017

# Table des matières

<b>1</b>	<b>Partie I : Implémentation des deux programme : voiture et client.</b>	<b>2</b>
1	Test avec $N = 4$ . . . . .	2
2	Test avec $N = 6$ . . . . .	2
3	Test avec $N = 8$ . . . . .	3
<b>2</b>	<b>Partie II : Conséquence de l'envoi d'un signale KILL à un processus Client et solution.</b>	<b>3</b>
4	Conséquence du KILL. . . . .	3
5	Solution en cas d'une action KILL. . . . .	4
<b>3</b>	<b>Partie III : Limitation du nombre de tournées pour chaque Client.</b>	<b>4</b>
6	Modification du code. . . . .	4
7	Contraintes. . . . .	5
7.1	Test avec un (1) groupe de client lancés dans un (1) terminal. . . . .	5
7.2	Test avec deux (2) groupes de client lancés dans deux (2) terminaux. . . . .	5
<b>4</b>	<b>Partie IV : Utilisation de la fonction Semtimedop.</b>	<b>6</b>
8	Problématique. . . . .	6
9	Solution. . . . .	6
9.1	Fonctionnement : . . . . .	6
10	Fonctionnalité avancée. . . . .	6
10.1	Implémentation : . . . . .	6
<b>5</b>	<b>Partie V : Implémentation de la barrière de synchronisation.</b>	<b>8</b>

# 1 Partie I : Implémentation des deux programme : voiture et client.

Les deux programmes C ont été implémenté,  
Des message texte indique les differents états du train ainsi que des clients,  
(aussi le déplacement du train est représenté graphiquement).

Puis testé avec quelques valeurs de N (le nombre de client) comme suit :

## 1 Test avec N = 4

```
izan@izan-Inspiron-5559: ~/Desktop/Naila/Sys/Parc
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ gcc client.c
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./client
groupe de sémaphore déjà créée et son id = 622605
Donner le nombre de clients : 4

---> Client : 7689 , je vais monter.
--- Client : 7689 , je suis en balade! ---
---> Client : 7690 , je vais monter.
--- Client : 7690 , je suis en balade! ---
---> Client : 7692 , je vais monter.
--- Client : 7692 , je suis en balade! ---
---> Client : 7691 , je vais monter.
--- Client : 7691 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ Client : 7689
la balade, je desends --->
Client : 7690 , FIN de la balade, je desends --->
Client : 7691 , FIN de la balade, je desends --->
Client : 7692 , FIN de la balade, je desends --->

```

```
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ gcc voiture.c -o voiture
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./voiture
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

| train \_
o-o-o-o-o-o-|

| train \_      ---->   | train \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|

| train \_      ---->   | train \_      ---->   | train \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|          o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2

```

## 2 Test avec N = 6

```
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./client
groupe de sémaphore déjà créée et son id = 622605
Donner le nombre de clients : 6

---> Client : 7762 , je vais monter.
--- Client : 7762 , je suis en balade! ---
---> Client : 7763 , je vais monter.
--- Client : 7763 , je suis en balade! ---
---> Client : 7764 , je vais monter.
--- Client : 7764 , je suis en balade! ---
---> Client : 7765 , je vais monter.
--- Client : 7765 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ Client : 7762
la balade, je desends --->
Client : 7763 , FIN de la balade, je desends --->
Client : 7765 , FIN de la balade, je desends --->
Client : 7764 , FIN de la balade, je desends --->
---> Client : 7767 , je vais monter.
--- Client : 7767 , je suis en balade! ---
---> Client : 7766 , je vais monter.
--- Client : 7766 , je suis en balade! ---

```

```
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ gcc voiture.c -o voiture
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./voiture
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

| train \_
o-o-o-o-o-o-|

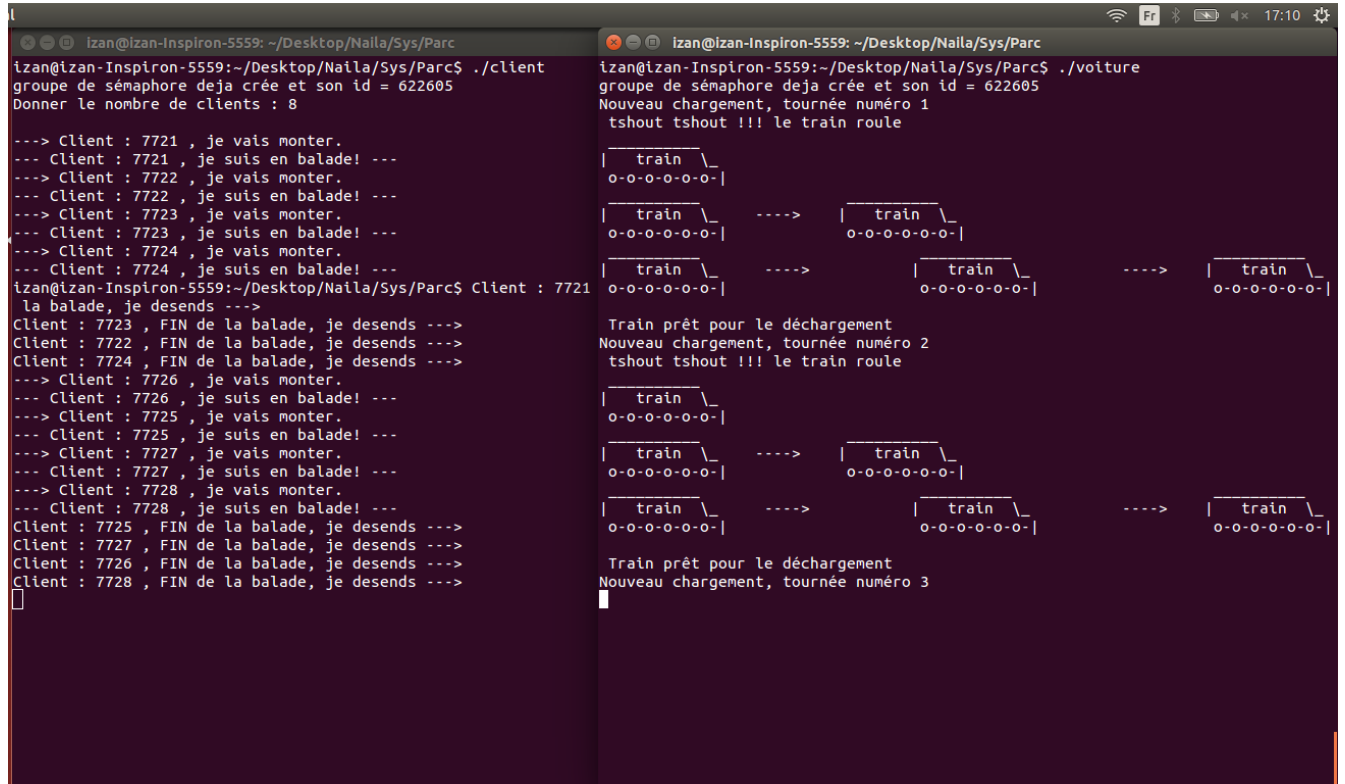
| train \_      ---->   | train \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|

| train \_      ---->   | train \_      ---->   | train \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|          o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2

```

### 3 Test avec $N = 8$



```
izan@izan-Inspiron-5559: ~/Desktop/Naila/Sys/Parc
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./client
groupe de sémaphore déjà crée et son id = 622605
Donner le nombre de clients : 8

---> Client : 7721 , je vais monter.
--- Client : 7721 , je suis en balade! ---
---> Client : 7722 , je vais monter.
--- Client : 7722 , je suis en balade! ---
---> Client : 7723 , je vais monter.
--- Client : 7723 , je suis en balade! ---
---> Client : 7724 , je vais monter.
--- Client : 7724 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ Client : 7721
la balade, je descends --->
Client : 7723 , FIN de la balade, je descends --->
Client : 7722 , FIN de la balade, je descends --->
Client : 7724 , FIN de la balade, je descends --->
---> Client : 7726 , je vais monter.
--- Client : 7726 , je suis en balade! ---
---> Client : 7725 , je vais monter.
--- Client : 7725 , je suis en balade! ---
---> Client : 7727 , je vais monter.
--- Client : 7727 , je suis en balade! ---
---> Client : 7728 , je vais monter.
--- Client : 7728 , je suis en balade! ---
Client : 7725 , FIN de la balade, je descends --->
Client : 7727 , FIN de la balade, je descends --->
Client : 7726 , FIN de la balade, je descends --->
Client : 7728 , FIN de la balade, je descends --->
█

izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./voiture
groupe de sémaphore déjà crée et son id = 622605
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

| train \_
o-o-o-o-o-o-|

| train \_      ----> | train \_
o-o-o-o-o-o-|      o-o-o-o-o-o-|

| train \_      ----> | train \_      ----> | train \_
o-o-o-o-o-o-|      o-o-o-o-o-o-|      o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2
tshout tshout !!! le train roule

| train \_
o-o-o-o-o-o-|

| train \_      ----> | train \_
o-o-o-o-o-o-|      o-o-o-o-o-o-|

| train \_      ----> | train \_      ----> | train \_
o-o-o-o-o-o-|      o-o-o-o-o-o-|      o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 3
█
```

## 2 Partie II : Conséquence de l'envoi d'un signal KILL à un processus Client et solution.

### 4 Conséquence du KILL.

Après avoir fait le "kill -9 pid" (avec ici pid = 7947)

On remarque que le client de pid = 7947 n'est pas descendu du train (il n'a pas affiché "Client 7947 , FIN de la balade, je descends —> ")

Donc il n'a pas libéré la ressource , c'est pourquoi on remarque que le train (voiture) ne permet pas de faire un autre voyage (elle n'affiche pas "Nouveau chargement, tournée numéro 2").

```

izan@izan-Inspiron-5559: ~/Desktop/Naila/Sys/Parc
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./client
groupe de sémaphore déjà créée et son id = 655374
Donner le nombre de clients : 4

---> Client : 7944 , je vais monter.
--- Client : 7944 , je suis en balade! ---
---> Client : 7945 , je vais monter.
--- Client : 7945 , je suis en balade! ---
---> Client : 7946 , je vais monter.
--- Client : 7946 , je suis en balade! ---
---> Client : 7947 , je vais monter.
--- Client : 7947 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ Client : 7945
la balade, je descends --->
Client : 7946 , FIN de la balade, je descends --->
Client : 7944 , FIN de la balade, je descends --->

```

```

izan@izan-Inspiron-5559:~/Desktop/Naila/Sys/Parc$ ./voiture
groupe de sémaphore déjà créée et son id = 655374
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

|  train  \_
o-o-o-o-o-o-|

|  train  \_      ----> |  train  \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|

|  train  \_      ----> |  train  \_      ----> |  train  \_
o-o-o-o-o-o-|          o-o-o-o-o-o-|          o-o-o-o-o-o-|

Train prêt pour le déchargement

```

```

izan@izan-Inspiron-5559:~$ kill -9 7947
izan@izan-Inspiron-5559:~$

```

## 5 Solution en cas d'une action KILL.

Le fait d'utiliser le flag "SEM\_UNDO" dans la fonction P du fichier semaphores2.h qu'on a créé, on peut observer que le kill -9 pid effectivement tue le processus client spécifié, mais le train continue de fonctionner normalement.

Tel que ce flag permet de libérer la ressource en cas de problème avec le processus bloquant. (il défait les actions du processus bloquant)

Ainsi on ne risque pas d'avoir d'attente (blocage) de durée indéterminée. mais cette dernière risque de causer des incohérences, selon la situation.

## 3 Partie III : Limitation du nombre de tournées pour chaque Client.

### 6 Modification du code.

Afin de limiter le nombre de tournées pour chaque client à un nombre déterminé "nb" par le processus voiture, On modifie le programme "voiture.c"

Tel qu'on lui donne en paramètre d'entrée ce nombre à l'exécution (exemple : ./voiture 2) pour que chaque client fasse 2 tournées.

Puis pour que les clients prennent en considération cette limitation, la valeur est envoyée vers une variable de la structure composant la mémoire partagée, qu'on a nommé : "nbTournéeMax"

Voici la nouvelle structure de shdata :

```

//structure de la mémoire partagée
typedef struct data{
    int nbEmbarques;
    int nbDebarques;
    int nbTournéeMax;
}shdata;

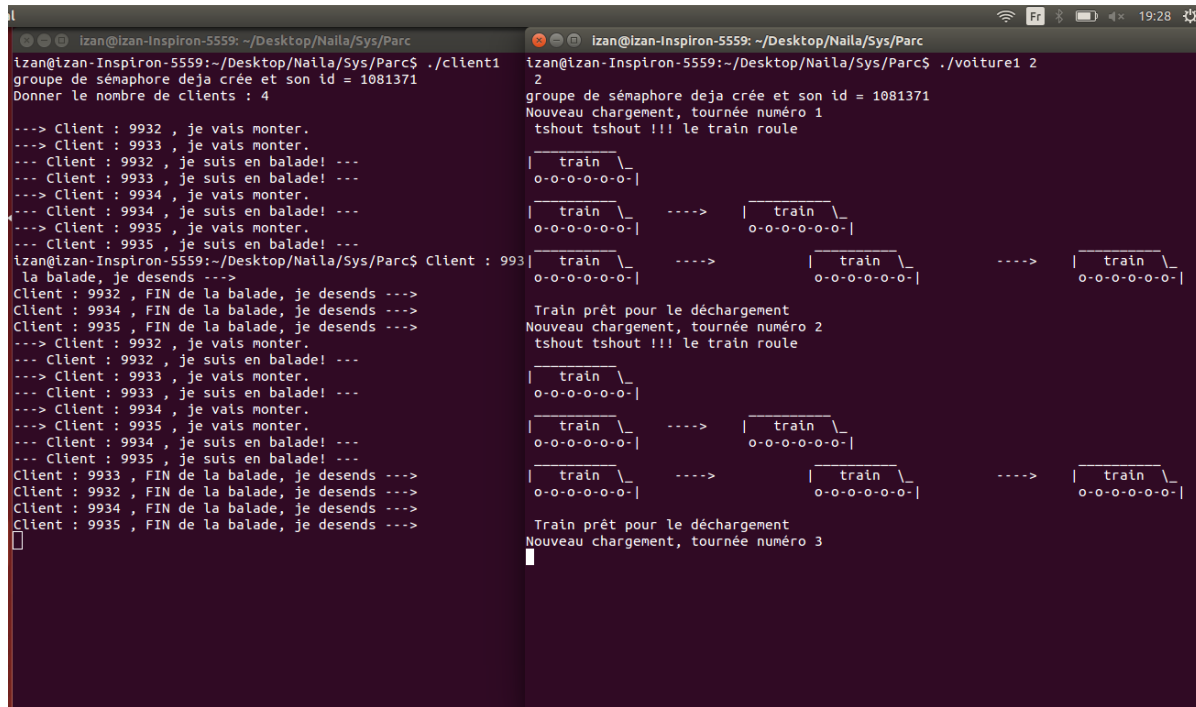
```

puis pour chaque client nous avons un compteur qui s'incrémente à chaque tournée effectuée. puis comparé avec le nombre maximum de tournée autorisé (lu dans nbTournéeMax) pour tester l'arrêt.

## 7 Contraintes.

Même après la fin de l'exécution d'un groupe de client via un terminal, si on lance un autre nombre de clients le processus de voiture (train) les prend en charge immédiatement.

### 7.1 Test avec un (1) groupe de client lancés dans un (1) terminal.



```
izan@izan-Inspiron-5559: ~/Desktop/Naïla/Sys/Parc$ ./client1
groupe de sémaphore déjà crée et son id = 1081371
Donner le nombre de clients : 4

---> Client : 9932 , je vais monter.
---> Client : 9933 , je vais monter.
--- Client : 9932 , je suis en balade! ---
--- Client : 9933 , je suis en balade! ---
---> Client : 9934 , je vais monter.
--- Client : 9934 , je suis en balade! ---
---> Client : 9935 , je vais monter.
--- Client : 9935 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ Client : 993
la balade, je descends --->
Client : 9932 , FIN de la balade, je descends --->
Client : 9934 , FIN de la balade, je descends --->
Client : 9935 , FIN de la balade, je descends --->
---> Client : 9932 , je vais monter.
--- Client : 9932 , je suis en balade! ---
---> Client : 9933 , je vais monter.
--- Client : 9933 , je suis en balade! ---
---> Client : 9934 , je vais monter.
--- Client : 9934 , je suis en balade! ---
---> Client : 9935 , je vais monter.
--- Client : 9935 , je suis en balade! ---
Client : 9933 , FIN de la balade, je descends --->
Client : 9932 , FIN de la balade, je descends --->
Client : 9934 , FIN de la balade, je descends --->
Client : 9935 , FIN de la balade, je descends --->
[]

izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ ./voiture1 2
2
groupe de sémaphore déjà crée et son id = 1081371
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2
tshout tshout !!! le train roule

| train \
0-0-0-0-0-0-|

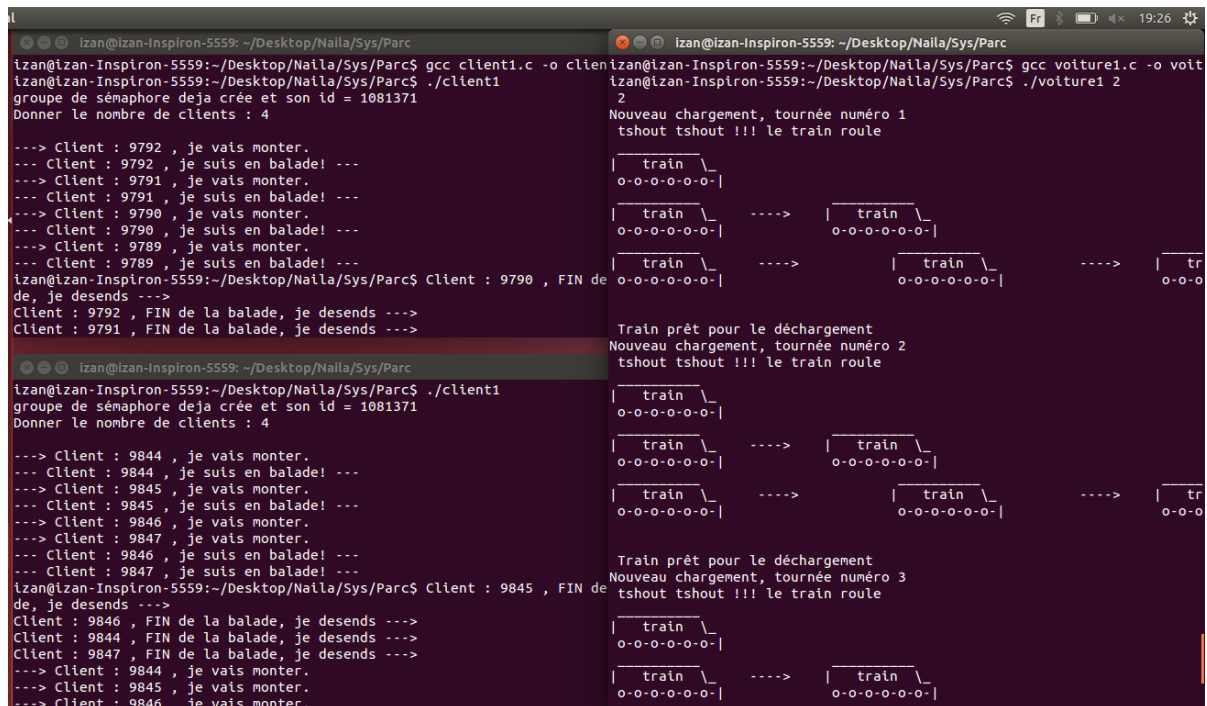
| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 3
```

### 7.2 Test avec deux (2) groupes de client lancés dans deux (2) terminals.



```
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ gcc client1.c -o client1
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ ./client1
groupe de sémaphore déjà crée et son id = 1081371
Donner le nombre de clients : 4

---> Client : 9792 , je vais monter.
--- Client : 9792 , je suis en balade! ---
---> Client : 9791 , je vais monter.
--- Client : 9791 , je suis en balade! ---
---> Client : 9790 , je vais monter.
--- Client : 9790 , je suis en balade! ---
---> Client : 9789 , je vais monter.
--- Client : 9789 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ Client : 9790 , FIN de
de, je descends --->
Client : 9792 , FIN de la balade, je descends --->
Client : 9791 , FIN de la balade, je descends --->

izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ ./client1
groupe de sémaphore déjà crée et son id = 1081371
Donner le nombre de clients : 4

---> Client : 9844 , je vais monter.
--- Client : 9844 , je suis en balade! ---
---> Client : 9845 , je vais monter.
--- Client : 9845 , je suis en balade! ---
---> Client : 9846 , je vais monter.
--- Client : 9846 , je suis en balade! ---
---> Client : 9847 , je vais monter.
--- Client : 9847 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ Client : 9845 , FIN de
de, je descends --->
Client : 9846 , FIN de la balade, je descends --->
Client : 9844 , FIN de la balade, je descends --->
Client : 9847 , FIN de la balade, je descends --->
---> Client : 9844 , je vais monter.
---> Client : 9845 , je vais monter.
---> Client : 9846 , je vais monter.

izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ gcc voiture1.c -o voiture1
izan@izan-Inspiron-5559:~/Desktop/Naïla/Sys/Parc$ ./voiture1 2
2
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2
tshout tshout !!! le train roule

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 3
tshout tshout !!! le train roule

| train \
0-0-0-0-0-0-|

| train \
0-0-0-0-0-0-|
```

## 4 Partie IV : Utilisation de la fonction Semsimedop.

### 8 Problématique.

S'il n'y a pas "p" client au minimum le train ne démarre pas , car c'est lorsque le dernier client (le p ième) monte dans le train que le démarrage de ce dernier est lancé (réveillé).

### 9 Solution.

La solution est d'utiliser la fonction "semsimedop" au lieu de "semop", pour cela nous avons ajouté à notre bibliothèque la fonction suivante :

```
void Ptimed(int semid , int n) {  
  
    struct sembuf sempar ;  
  
    sempar.sem_num = n ;  
    sempar.sem_op = -1 ;  
    sempar.sem_flg = 0;  
  
    struct timespec timeout;  
  
    timeout.tv_sec = 7;  
    timeout.tv_nsec = 0;  
  
    semsimedop(semid,&sempar,1,&timeout);  
}
```

#### 9.1 Fonctionnement :

semsimedop() : permet de délimiter le temps d'attente d'un processus endormi(verrouillé avec P)  
Tel que si le processus dépasse le timeout donné en argument a cette fonction, l'attente sera interrompue, donc le processus bloqué sera réveillé.

dans notre exemple du park, lorsque le train (non pleine) aura attendu un certain temps (toujours non plein) il va quand même faire le voyage.

PS : si le timeout donné est NULL alors le temps attente sera infini , se qui nous ramènera a un résultat identique a celui de la fonction sem\_op()

## 10 Fonctionnalité avancée.

### 10.1 Implémentation :

La fonction "semsimedop()" a réglé le problème précédent :  
Maintenant si le train n'est toujours pas rempli après un certain temps (7 seconds) il démarre et fait le voyage.

Le programme voiture.c a aussi été modifié pour ne démarré que si le nombre de client a bord est supérieur au nombre de place divisé par 3.

dans notre cas nous avons pris :

p = 4 (4 places dans le train)

Ainsi le train ne démarre que s'il y'a au moins 2 clients a bord du train (nbClient  $\geq$  p/3)

Au début aucune client (nbClient = 0) train annulé.

puis 2 clients montent , Le train roule.

puis 1 client monte , Le train est annulé.

```
izan@izan-Inspiron-5559: ~/Desktop/Naila/t
izan@izan-Inspiron-5559:~/Desktop/Naila/t$ gcc voiture1.c -o voiture1
izan@izan-Inspiron-5559:~/Desktop/Naila/t$ ./voiture1 1
groupe de sémaphore déjà créée et son id = 1245209
Nouveau chargement, tournée numéro 1
nbClient en attente = 0

***** train annulé !!! pas assés de passagers *****

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 2
nbClient en attente = 2
tshout tshout !!! le train roule

|  train  \_
|  o-o-o-o-o-o-|
|  train  \_      ----> |  train  \_
|  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|
|  train  \_      ----> |  train  \_      ----> |  train  \_
|  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 3
nbClient en attente = 1

***** train annulé !!! pas assés de passagers *****

Train prêt pour le déchargement
█
```

En suite 4 clients , Le Train est plein il démarre.

```
izan@izan-Inspiron-5559: ~/Desktop/Naila/t
|  train  \_      ----> |  train  \_      ----> |  train  \_
|  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 3
nbClient en attente = 1

***** train annulé !!! pas assés de passagers *****

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 4
nbClient en attente = 0

***** train annulé !!! pas assés de passagers *****

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 5
nbClient en attente = 0

***** train annulé !!! pas assés de passagers *****

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 6
nbClient en attente = 4
tshout tshout !!! le train roule

|  train  \_
|  o-o-o-o-o-o-|
|  train  \_      ----> |  train  \_
|  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|
|  train  \_      ----> |  train  \_      ----> |  train  \_
|  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|      |  o-o-o-o-o-o-|

Train prêt pour le déchargement
Nouveau chargement, tournée numéro 7
nbClient en attente = 0
█
```

```
izan@izan-Inspiron-5559: ~/Desktop/Naila/t
izan@izan-Inspiron-5559:~/Desktop/Naila/t$ gcc client1.c -o client1
izan@izan-Inspiron-5559:~/Desktop/Naila/t$ ./client1
groupe de sémaphore déjà créée et son id = 1245209
Donner le nombre de clients : 2

izan@izan-Inspiron-5559:~/Desktop/Naila/t$ ---> Client : 30171 , je vais monter.
---> Client : 30170 , je vais monter.
--- Client : 30171 , je suis en balade! ---
--- Client : 30170 , je suis en balade! ---
Client : 30171 , FIN de la balade, je descends --->
Client : 30170 , FIN de la balade, je descends --->
./client1
groupe de sémaphore déjà créée et son id = 1245209
Donner le nombre de clients : 1

izan@izan-Inspiron-5559:~/Desktop/Naila/t$ ---> Client : 30173 , je vais monter.
--- Client : 30173 , je suis en balade! ---
Client : 30173 , FIN de la balade, je descends --->
./client1
groupe de sémaphore déjà créée et son id = 1245209
Donner le nombre de clients : 4

izan@izan-Inspiron-5559:~/Desktop/Naila/t$ ---> Client : 30200 , je vais monter.
---> Client : 30199 , je vais monter.
--- Client : 30200 , je suis en balade! ---
--- Client : 30199 , je suis en balade! ---
---> Client : 30202 , je vais monter.
--- Client : 30202 , je suis en balade! ---
---> Client : 30201 , je vais monter.
--- Client : 30201 , je suis en balade! ---
Client : 30200 , FIN de la balade, je descends --->
Client : 30199 , FIN de la balade, je descends --->
Client : 30202 , FIN de la balade, je descends --->
Client : 30201 , FIN de la balade, je descends --->
```



## 5 Partie V : Implémentation de la barrière de synchronisation.

La barrière de synchronisation implémentée prend en paramétrés :

semid : identifiant unique du groupe de sémaphore sur le quel on veut agir.

n1 : le numéro du sémaphore sur le quel on veut faire un P.

n2 : le numéro du sémaphore sur le quel on veut faire un V.

nb : le numéro du processus à mettre en attente.

nbProcessus : le nombre total de processus communiquant.

```
void Synchro(int semid , int n1 , int nb, int nbProcessus , int n2)
{
    int i;

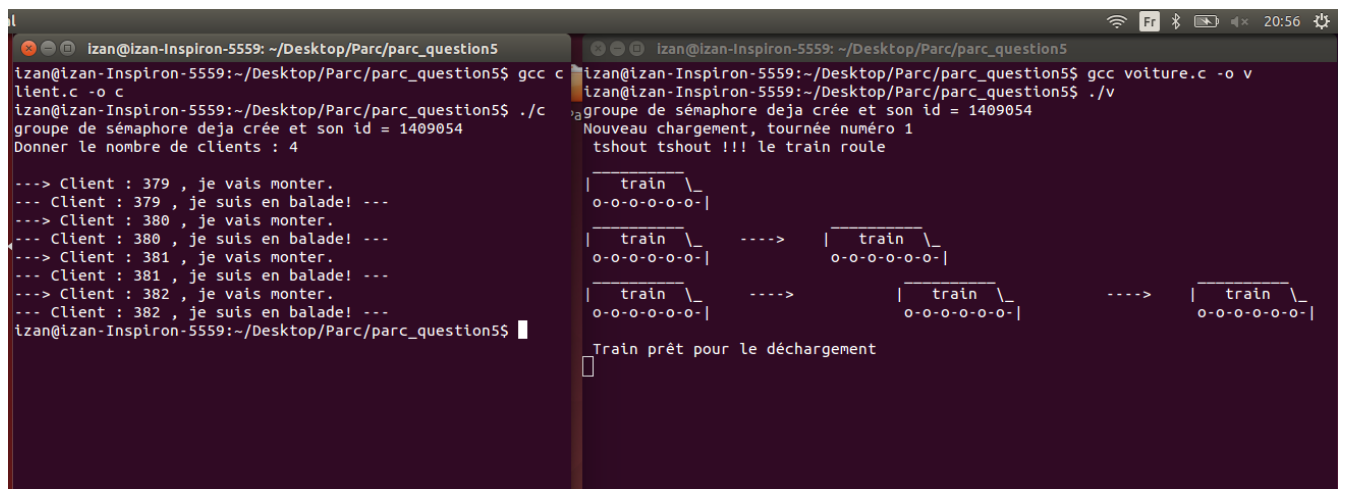
    if(nb < nbProcessus-1)
    {
        P(semid , n1);
    }
    else
    {
        for(i = 0 ; i < nbProcessus ; i++)
        {
            V(semid , n2);
        }
    }
}
```

Ainsi on test si le numéro du processus courant (nb) n'est pas le dernier alors on applique P (mise en attente). Sinon (c'est le dernier) on applique à tous les processus précédents (dans une boucle) un V (libération).

Cette fonction "Synchro" a été ajoutée à la bibliothèque "semaphores2.h"

Et fait appel aux deux fonctions déjà créées P et V.

Pour la tester nous l'avons intégrée dans l'exemple du parc d'attraction.  
et voici le résultat :



```
izan@izan-Inspiron-5559: ~/Desktop/Parc/parc_question5
izan@izan-Inspiron-5559:~/Desktop/Parc/parc_question5$ gcc c
lient.c -o c
izan@izan-Inspiron-5559:~/Desktop/Parc/parc_question5$ ./c
groupe de sémaphore déjà créée et son id = 1409054
Donner le nombre de clients : 4
---> Client : 379 , je vais monter.
--- Client : 379 , je suis en balade! ---
---> Client : 380 , je vais monter.
--- Client : 380 , je suis en balade! ---
---> Client : 381 , je vais monter.
--- Client : 381 , je suis en balade! ---
---> Client : 382 , je vais monter.
--- Client : 382 , je suis en balade! ---
izan@izan-Inspiron-5559:~/Desktop/Parc/parc_question5$
```

```
izan@izan-Inspiron-5559:~/Desktop/Parc/parc_question5$ gcc voiture.c -o v
izan@izan-Inspiron-5559:~/Desktop/Parc/parc_question5$ ./v
groupe de sémaphore déjà créée et son id = 1409054
Nouveau chargement, tournée numéro 1
tshout tshout !!! le train roule

  |  train  \_
o-o-o-o-o-o-|

  |  train  \_      ---->  |  train  \_
o-o-o-o-o-o-|              o-o-o-o-o-o-|

  |  train  \_      ---->  |  train  \_      ---->  |  train  \_
o-o-o-o-o-o-|              o-o-o-o-o-o-|              o-o-o-o-o-o-|

Train prêt pour le déchargement
□
```