Université des Sciences et de la Technologie Houari Boumediene

Systemes d'Exploitations

Mini Projet : Threads, Sémaphores, Shared memories

Binôme
MOHAMMEDI Haroune
KADRI Adlane

Professeur
Pr. MEHDI Malika

31 décembre 2017

1 Exercice du parc d'attraction

1.1 Question 1

1.1.1 Implementaion

Pour l'implémentation du problème donnée en utilisant le système `IPC V`, on a définit les sémaphores indiqué dans la solution théorique donnée à l'aide des appels système linux `semget, semctl` et une mémoire partagée pour sauvegarder le nombre de clients embarquées et débarquées (la structure `state`). les mémoires partagées sont gérées à l'aide des appels système `shmget, shmat`. Notant que l'utlisation de ces appels système nécessite la création d'un clé `key_t`.

Pour facilité la tache de compilation des différents fichiers **C** on a utlisé l'outil **CMake**. pour cela on écrit le fichier `CMakeLists.txt` qui contient tout les information concernant les fichier du project.

1.1.2 Exécution avec `N = 4` et `P = 3`

Pour exécuter `P` processus on a écrit un programme `run_clients` qui prends `N` comme paramètre du `main` et qui `N` processus clients en utlisant les fonctions `fork, execlp`.

Voici les sorties de l'exécution des processus :

Listing 1 – Voiture.c

```
groupe de sema deja creer et son id est : 65536
Initialisation des sémaphores :
mutex1          --> 1
mutex1          --> 1
semEmbarquement --> 0
semDebarquement --> 0
semTousAbord    --> 0
semTousDehors   --> 0
Segment memoire existe deja id:18776071
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 1
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 2
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 3
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 4
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 5
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 6
processus voiture entraine de rouler
```

Listing 2 – run_clients.c

```
Usage : run_clients nb_clients
client 24030: je vais monter
client 24029: je vais monter
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 24030: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
client 24029: je suis en balade
```

```
client 24030: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24029: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
client 24032: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
client 24030: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 24030: je suis en balade
client 24031: je vais monter
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 24031: je suis en balade
client 24029: je vais monter
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
client 24029: je suis en balade
client 24030: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24031: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
client 24029: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
client 24032: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 24032: je suis en balade
client 24031: je vais monter
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 24031: je suis en balade
client 24030: je vais monter
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
client 24030: je suis en balade
client 24030: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24032: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24031: FIN de la balade je descend
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
client 24032: je vais monter
client 24029: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 24032: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 24029: je suis en balade
client 24030: je vais monter
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
```

```
client 24030: je suis en balade
client 24030: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24032: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
client 24029: FIN de la balade je descend
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
client 24031: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 24031: je suis en balade
client 24030: je vais monter
client 24032: je vais monter
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 24030: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
client 24032: je suis en balade
```

1.2 Question 2

1.2.1 Problème

l'envoie un signal kill a un des processus client pendant que la voiture est en tournée (commande : `kill -9 pid`) bloque le processuss voiture et tout autres processus clients dans le systems. le blockage est dû au vaiable `nbDebarques` qui ne sera jamais égales à `P` et donc le processus voiture se block dans l'instruction `P(semid, semTousDehors)` et les processus clients se block dans l'instruction `P(semid, semEmbarquement)`.

1.2.2 Utilisation de `SEM_UNDO`

le flag `SEM_UNDO` permet au processus de libérer les sémaphores dans le quelle il a l'acces avant sa terminision (remetre en cause les différentes modifications sur les différentes sémaphores) en utilisant une variable special `semadj` qui est définie pour chaque processus et pour chaque semaphore et qui sauvgarde la somme de toutes les operations effectuer par chaque processus sur chaque sémaphore.

Avec l'utlisation de ce flag dans les opérations des sémaphores `P` et `V` le problème ne sera pas réglé puisque il été engendré par la valeur du variable `nbDebarques` non pas par des valuers de sémaphores.

1.2.3 Solution Propsée

On peut réglé le problème si on utilise un sémaphore `semDedans` qui sera incrémenté a chaque embarquement d'un processus et décrementé à chaque débarquement, comme ça si on tue un processus qui a monté et qui n'a pas encore decendu (c'est la cause du problème précedent) la valeur du sémaphore `semDedans` va décrementer en specifiant le flag `SEM_UNDO`, et donc le processus voiture ne va pas bloquer puisque le nombre des clients dans la voiture

Détail sur `SEM_UNDO`

- si la valeur du sémaphore est modifié par `SETVALL` ou `SETALL` la valeur du variable special `semadj` est reinitialisé. On doit pensé à utilisé les flags du système `CLONE_SYSTEMV` dans ce cas.

- l'OS linux applique la stratégie **décrementer la valeur tant que possible** lors du remetre en cause des operations effectuer par un processus sur un sémaphore (effet du flag `SEM_UNDO`).

1.3 Question 3

1.3.1 Implementaion

Pour l'implémentation du limite des trounée pour chaque clients, on doit sauvegarder le nombre du tournée pour chaque clients. Pour cela on a définie les variables `int nbClients`, `int clients[]` et `int tours[]` dans la structure `state`. `nbClients` contients le nombre de clients dans le systèmes. `clients` contients les **pids** des clients dans le système. `tours` contients le nombre de tournées pour chaque **pid** dans `clients`.

on définie aussi les variables `MAX_TOURS` qui peut être mise à jourer par le processus voiture et `MAX_CLIENTS` qui contient le nombre maximum du clients qui peut être enregister dans les tableaux précedents.

Une sémahpore `mutex3` pour protéger les variables `int nbClients`, `int clients[]` et `int tours[]`

Note : une milleur solution peut etre implementer en utilisant les structure dynamiques.

L'idée est que chaque nouveau clients (son **pid**) est sauvegarder dans la case numéro `nbClients` du tableau **clients** et son nombre de tournée associer est sauvegardé dans la case numéro `nbClients` du tableau `tours`.

A l'arrivé du clients il fait appele à la foction `inscription` qui lui enregister dans le tableau `clients` et initialise son nombre de tours à zero.

Pour pouvoir fait une tournée (la condition d'entrée) il fait appele à la fonction `peutTourner` qui retourne `vrai` que si le nombre de tours du clients est inférieur au nombre maximale de trournée

Chaque fin de tournées le processus cliens fait appele à la fonction `finTour` qui incrémente son nombre de tours.

1.3.2 Exécution avec `N = 4` et `P = 3`

Pour exécuter `P` processus on a écrit un programme `run_clients` qui prends `N` comme paramètre du `main` et qui crée `N` processus clients en utlisant les fonctions `fork, execlp`.

Voici les sorties de l'exécution des processus :

Listing 3 – Processus Voiture

```
groupe de sema deja creer et son id est : 0
Initialisation des sémaphores :
mutex1          --> 1
mutex2          --> 1
mutex3          --> 1
semEmbarquement --> 0
semDebarquement --> 0
semTousAbord    --> 0
semTousDehors   --> 0
Segment memoire existe deja id:1572869
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 1
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 2
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 3
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
nouveau chargement tournée numéro 4
```

1.4 Question 4

1.4.1 Pricipe du `semtimedop`

`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 `timespec` dont l'adresse est transmise dans le paramètre timeout. Si la limite indiquée a été atteinte, `semtimedop()` échoue avec errno contenant `EAGAIN` (et aucune opération de sops n'est réalisée).

1.4.2 Solution Théorique

l'utilisation d'une fonction `Ptimed` au lieu de `P` dans le processus voiture avec le sémaphore `semTousAbord` nous permet de garantir une attente borné dans la file du sémaphore en question. Après la libération le processus voiture remet a zéro la valeur du variable `nbEmbarques` et du sémaphore `semEmbarquemnt` (appeles successifs du `V` pour bloquer les eventueles clients qui arrivent). Il vérifie si il ya plus de `C / 3` client dans la variable `nbEmbarques` si oui il commence a tourner (fonction `rouler()`) sinon il décharge les clients et tente une autre fois.

lors du déchargement on fait la meme chose avec la variable `nbDebarques` et les sémaphores `semDebarquement` et `semTousDehors`.

1.4.3 Implementaion

la valeur `EAGAIN` dans `errno` nous permet de savoir si la terminison du fonction `semtimedop` est dû à l'écoulement du temps spécifiée dans `timeout`, les valeurs des variables et des sémaphores sont réinitialisées à l'aide des fonctions déffine dans notre librarie `semaphore1.h` notament `setNbDebarques`, `setNbEmbarques` et `V(semid, id`.

1.4.4 Resultats de l'execution Avec 0 clients puis avec 2

Listing 4 – Processus Voiture

```
groupe de sema deja creer et son id est : 32769
Initialisation des sémaphores :
mutex1          --> 1
mutex2          --> 1
semEmbarquement --> 0
semDebarquement --> 0
semTousAbord    --> 0
semTousDehors   --> 0
Segment memoire existe deja id:2424841
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 1
Chargmemet : temps d'attente 5 secondes ecoulé
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
processus voiture prêt pour le déchargement
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
calling Ptimed(semid, semTousDehors, timeout)Déchargement: temps d'attente 5 secondes
    ↪ ecoulé
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 2
Chargmemet : temps d'attente 5 secondes ecoulé
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
tourner quand même avec 2 clients
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
calling Ptimed(semid, semTousDehors, timeout)Déchargement: temps d'attente 5 secondes
    ↪ ecoulé
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 3
Chargmemet : temps d'attente 5 secondes ecoulé
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
tourner quand même avec 2 clients
processus voiture entraine de rouler
```

```
processus voiture prêt pour le déchargement
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
calling Ptimed(semid, semTousDehors, timeout)Déchargement: temps d'attente 5 secondes
    ↪ ecoulé
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
nouveau chargement tournée numéro 4
Chargmemet : temps d'attente 5 secondes ecoulé
lecture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
tourner quand même avec 2 clients
processus voiture entraine de rouler
processus voiture prêt pour le déchargement
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
calling Ptimed(semid, semTousDehors, timeout)
```

Listing 5 – Processus Clients

```
groupe de sema deja creer et son id est : 32769
Segment memoire existe deja id:2424841
groupe de sema deja creer et son id est : 32769
Segment memoire existe deja id:2424841
client 5634: je vais monter
client 5635: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 5634: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 5635: je suis en balade
client 5634: FIN de la balade je descend
client 5635: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
client 5635: je vais monter
client 5634: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 5635: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 5634: je suis en balade
client 5635: FIN de la balade je descend
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
client 5634: FIN de la balade je descend
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
client 5634: je vais monter
client 5635: je vais monter
lecture nbEmbarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
client 5634: je suis en balade
lecture nbEmbarques: State : nbEmbarques = 1, nbDebarques = 0
ecriture nbEmbarques: State : nbEmbarques = 2, nbDebarques = 0
client 5635: je suis en balade
client 5634: FIN de la balade je descend
client 5635: FIN de la balade je descend
```

```
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 0
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
lecture nbDebarques: State : nbEmbarques = 0, nbDebarques = 1
ecriture nbDebarques: State : nbEmbarques = 0, nbDebarques = 2
```

1.5 Question 5

1.5.1 Solution Théorique

La solution proposé est d'écrire une fonction `join(n)` qui appele `P` sur un sémaphore (c'est elle qui va le crée lors du premier appele) pour les premiers (N - 1) processus sauf le dernier pour le quelle elle appellera `V` en boucle pour réveiller tout les (N - 1) processus bloqués.

1.5.2 Implementaion

Pour savoir le nombre de nombre de processus qui attend dans la file d'un sémaphore donée on a implementé une fonction `getNConf` qui fait appele à la fonction `semctl` avec le flag `GETNCNT`

La fonction `join(n)` avec l'utlisation de la fonction `semget` avec les flags `IPC_CREATE | IPC_EXCL | 0666` vérifie si un sémaphore est créer. si oui elle vérifie la valuer de sémaphore (fonction `getNConf`) si la valeur est inférieur à **N** elle appele `V` en boulce sur le sémaphore crée précedement sinon elle `P` pour bloquer le processus. dans le cas où le groupe du sémaphore est n'est pas créer (`semget` retourne **-1**) la fonction créer un et l'initialise à **0**.

1.5.3 Resultats de l'execution avec `N = 3`

Listing 6 – Processus 1

```
waiting in the barrier
all the process came, I'm free now
```

Listing 7 – Processus 2

```
waiting in the barrier
all the process came, I'm free now
```

Listing 8 – Processus 3

```
last process, freeing all others
all the process came, I'm free now
```