Programmation temps réel
Projet Trains

LIN Jiangning 17809146
CALMEL Morvan 12314763

Dans les parties 1, 2 et 3, nous avons créé 3 threads, pour chaque trains (TrainUn, TrainDeux, TrainTrois)

Partie 1 - mutex :
Pour chaque train un mutex bloquant pour les 2 autres threads / trains en fonctionnement qui arrive en sens inverse. Pour éviter que 2 trains ne se rencontrent sur la même voie.

Partie 2 - sémaphore :
Trois sémaphores bloquant pour empêcher d'avoir un train qui arrive en sens inverse.

Partie 3 - Verrous
Trois verrous bloquant pour empêcher d'avoir un train qui arrive en sens inverse.

Partie 4 - Files de messages
Nous avons créé un thread pour chaque trains et deux fichiers (reader.c et writer.c)
Pour chaque train, ça affiche l'ordre des trajets, un train va faire un trajet, les deux autres doivent attendre. Tous les messages fait dans le fichier writer sont envoyés au fichier reader.

La partie1 s'inspire de la variante pour 3 groupes de lecteurs : Autre cas similaire équivalent à 3 groupe de “lecteurs”: trois groupes de trains qui doivent emprunter une voie en sens opposé. Plusieurs trains peuvent passer dans un sens, mais les trains arrivent dans l’ordre des départs.

Nous avons fait le modèle des mutex :

- On crée quatre mutex en déclarant une variable pthread_mutex_t.
- On déclare le temps en utilisant clock_t et colck() pour calculer le temps de chaque trajet et chaque tour.
- On déclare trois nombres des trains dans la voie, on permet à un unique train de circuler
- On déclare des éléments qui sont par exemple debut_train et fin_train sont deux gares du trajet qui a fini, possible_debut_train et possible_fin_train sont deux gares du trajet qui vont être utiliser.
- On déclare trois éléments pour permettre de limiter le calcul de la moyenne à trois fois.
- On crée une fonction char *substring(size_t start, size_t stop, const char *src, char *dst, size_t size) pour récupérer le caractère du début et la fin de la chaine de caractère, par exemple A --> B qui donne A et B.
- On crée une fonction int train_en_inverse(char *possible_debut1, char* possible_fin1, char* possible_debut2, char* possible_fin2) pour décider si deux trains sont en sens inverse, par exemple le Train1 A -->B, le Train2 B-->A
- On crée une fonction int train_en_ligne(char *possible_debut1,char* possible_fin1,char* possible_debut2, char* possible_fin2) pour décider si deux trains sont dans le même sens, par exemple le Train1 et Train2 doit faire A-->B.
- On créé trois fonctions void* TrainUn(void *p),void* TrainDeux(void *p), void* TrainDeux(void *p) pour faire les actions des trois trains. Sur chaque voie ferrée, il n'est possible qu'à un unique train de circuler. Si un train va de A vers B, alors aucun train ne peut aller de B vers A. Par contre, si un train va de A vers B, alors d'autres trains peuvent aller de A vers B. Pour un trajet, les trains arrivent dans l'ordre des départs. Donc dans chaque fonction qui contient une méthode pour verrouiller les deux autres trains dans les directions en inverse ou les mêmes trajets, on a les deux autres trains qui sont déverrouillés quand le train a fini le trajet. en utilisant une méthode strcmp() chercher un trajet s’il est la dernier trajet en récupérant le temps moyen d’un trajet complet. on déclare les trajets de trains, par exemple char train1[4][8] = {"A --> B", "B --> C", "C --> B", "B --> A"}. Nous avons simuler les temps de déplacement sur chaque liaison par un temps d’attente aléatoire de 1 à 3 secondes en utilisant sleep(rand()%3+1).
- Nous avons choisi les trois trains pour représenter trois threads, ils sont déclarés dans la fonction main().

et on utilise les fonctions pour réussir la partie1 ci-après:

```
ressource_type *ressource;
int nb_train1 = 0;
int nb_train2 = 0;
int nb_train3 = 0;

mutex mutex1 = 1; //pour train1
mutex mutex2 = 1; //pour train2
mutex mutex3 = 1; //pour train3
mutex fifo = 1;

TrainUn() {
        P(fifo); // file d'attente du Train1 et Train2
        P(mutex1);
        nb_train1 = nb_train1 + 1;
        if(nb_train1== 1){
                //compétition avec les autres train(le premier gagne)
                //on verrouille sous les cas des directions en inverse
                P(mutex2);
                p(mutex3);
        }
        V(mutex1);
```

```
        V(fifo);
        TrainUn(ressource);
        P(mutex1);
        nb_train1= nb_train1 - 1;
        if(nb_train1 == 0){
                V(mutex2);
                V(mutex3);
        }
        V(mutex1);
}
TrainDeux() {
        P(fifo); // file d'attente du Train1 et Train3
        P(mutex2);
        nb_train2 = nb_train2 + 1;
        if(nb_train2== 1){
                //compétition avec les autres train(le premier gagne)
                //on verrouille sous les cas des directions en inverse
                P(mutex1);
                p(mutex3);
        }
        V(mutex2);
        V(fifo);
        TrainDeux(ressource);
        P(mutex2);
        nb_train2= nb_train2 -1;
        if(nb_train2 == 0){
                V(mutex1);
                V(mutex3);
        }
        V(mutex2);
}
TrainTrois() {
        P(fifo); // file d'attente du Train1 et Train2
        P(mutex3);
        nb_train3 = nb_train3 + 1;
        if(nb_train3== 1){
                //compétition avec les autres train(le premier gagne)
                //on verrouille sous les cas des directions en inverse
                P(mutex1);
                p(mutex2);
        }
        V(mutex3);
        V(fifo);
        TrainTrois(ressource);
        P(mutex3);
        nb_train3= nb_train3 -1;
        if(nb_train3 == 0){
                V(mutex1);
                V(mutex2);
        }
        V(mutex3);
}
```

Compiler notre programme avec les options -Wall et -pthread :
$ **gcc -Wall partie1.c -o -pthread**
$ **./a.out**

un résultat ci- après:
//compétition avec les autres train(le premier gagne)
Le train1 doit aller le trajet: A --> B
1.902000 - Le train3 doit faire le trajet: A --> B
Le train1 fait le trajet: A --> B
Le train1 est parti de la gare: A

1.965000 - Le train1 a fini le trajet : A --> B
Le train1 est arrivé à la gare:B

1.989000 - Le train3 fait le trajet: A --> B
Le train3 est parti de la gare: A

2.002000 - Le train3 a fini le trajet : A --> B
Le train3 est arrivé à la gare:B

Le train2 doit faire le trajet: A --> B
2.063000 - Le train2 fait le trajet: A --> B
Le train2 est parti de la gare: A

2.081000 - Le train2 a fini le trajet : A --> B
Le train2 est arrivé à la gare:B

Le train1 doit aller le trajet: B --> C
2.179000 - Le train1 fait le trajet: B --> C
Le train1 est parti de la gare: B

2.210000 - Le train1 a fini le trajet : B --> C
Le train1 est arrivé à la gare:C

Le train3 doit faire le trajet: B --> D
2.346000 - Le train3 fait le trajet: B --> D
Le train3 est parti de la gare: B

2.386000 - Le train3 a fini le trajet : B --> D
Le train3 est arrivé à la gare:D

Le train1 doit aller le trajet: C --> B
2.442000 - Le train1 fait le trajet: C --> B
Le train1 est parti de la gare: C

2.459000 - Le train1 a fini le trajet : C --> B
Le train1 est arrivé à la gare:B

Le train1 doit aller le trajet: B --> A
2.501000 - Le train1 fait le trajet: B --> A
Le train1 est parti de la gare: B

2.514000 - Le train1 a fini le trajet : B --> A
Le train1 est arrivé à la gare:A

Le train3 doit faire le trajet: D --> C
2.537000 - Le train3 fait le trajet: D --> C
Le train3 est parti de la gare: D

2.547000 - Le train3 a fini le trajet : D --> C
Le train3 est arrivé à la gare:C

Le temps moyen du trajet du train 1 est de : 2.570000 secondes

Le train1 doit aller le trajet: A --> B
2.592000 - Le train1 fait le trajet: A --> B
Le train1 est parti de la gare: A

2.602000 - Le train1 a fini le trajet : A --> B
Le train1 est arrivé à la gare:B

Le train2 doit faire le trajet: B --> D
2.699000 - Le train2 fait le trajet: B --> D
Le train2 est parti de la gare: B

2.729000 - Le train2 a fini le trajet : B --> D
Le train2 est arrivé à la gare:D

Le train3 doit faire le trajet: C --> E
2.884000 - Le train3 fait le trajet: C --> E
Le train3 est parti de la gare: C

2.915000 - Le train3 a fini le trajet : C --> E
Le train3 est arrivé à la gare:E

Le train1 doit aller le trajet: B --> C
2.980000 - Le train1 fait le trajet: B --> C
Le train1 est parti de la gare: B

3.008000 - Le train1 a fini le trajet : B --> C
Le train1 est arrivé à la gare:C

Le train3 doit faire le trajet: E --> A
3.046000 - Le train3 fait le trajet: E --> A
Le train3 est parti de la gare: E

3.058000 - Le train3 a fini le trajet : E --> A
Le train3 est arrivé à la gare:A

Le train1 doit aller le trajet: C --> B
3.111000 - Le train1 fait le trajet: C --> B
Le train1 est parti de la gare: C

3.128000 - Le train1 a fini le trajet : C --> B
Le train1 est arrivé à la gare:B

Le temps moyen du trajet du train 3 est de : 3.147000 secondes
……

Pour la partie 2, on utilise les sémaphores.
On a déclaré 5 fonctions , une pour la récupération du caractère de début et de fin d'un trajet.
3 fonctions pour chacun des 3 threads. Et la fonction main.
On récupère les trains en sens inverse et on ajoute un sémaphore bloquant qui est sensé bloquer l'accès à la voie.

Dont voici un résultat après exécution :
1.392000 - train 3 en approche : A --> B
Train 1 en approche : A --> B
1.368000 - train 2 en approche : A --> B
1.454000 - Le train 1 est arrivé à la gare : B

1.464000 - Train 1 en approche : B --> C
1.468000 - Le train 2 est arrivé à la gare : B

1.480000 - train 2 en approche : B --> D
1.516000 - Le train 2 est arrivé à la gare : D

1.581000 - train 2 en approche : D --> C
1.625000 - Le train 3 est arrivé à la gare : B

1.702000 - train 3 en approche : B --> D
1.725000 - Le train 1 est arrivé à la gare : C

1.789000 - Train 1 en approche : C --> B
1.839000 - Le train 1 est arrivé à la gare : B

Le temps moyen du train 1 est de : 1.906000 secondes

1.924000 - Train 1 en approche : B --> A
1.949000 - Le train 2 est arrivé à la gare : C

1.963000 - train 2 en approche : C --> B
1.979000 - Le train 1 est arrivé à la gare : A

1.996000 - Train 1 en approche : A --> B
2.035000 - Le train 3 est arrivé à la gare : D

2.103000 - train 3 en approche : D --> C
2.149000 - Le train 2 est arrivé à la gare : B
Le temps moyen du train 2 est de : 2.216000 secondes

2.276000 - Le train 1 est arrivé à la gare : B

2.292000 - Train 1 en approche : B --> C
2.331000 - Le train 3 est arrivé à la gare : C

2.398000 - train 3 en approche : C --> E
2.424000 - Le train 3 est arrivé à la gare : E

Le temps moyen du train 3 est de : 2.444000 secondes

Dans la partie 3, on utilise les verrous.
On a déclaré 5 fonctions , une pour la récupération du caractère de début et de fin d'un trajet.
3 fonctions pour chacun des 3 threads. Et la fonction main.
On récupère les trains en sens inverse et on ajoute un verrou qui est sensé bloquer l'accès à la voie.

Voici un retour du programme :

1.000000 - Train 1 : A --> B
1.000000 - 1.000000 - Train 2 : A --> B
Train 3 : A --> B
1.894000 - Le train 3 est arrivé à la gare : B

1.000000 - Train 3 : B --> D
1.943000 - Le train 1 est arrivé à la gare : B

1.968000 - Le train 2 est arrivé à la gare : B

1.987000 - Le train 3 est arrivé à la gare : D

2.000000 - Train 2 : B --> D
2.000000 - Train 3 : D --> C
2.000000 - Train 1 : B --> C
2.356000 - Le train 1 est arrivé à la gare : C

2.000000 - Train 1 : C --> B
2.454000 - Le train 3 est arrivé à la gare : C

2.000000 - Train 3 : C --> E
2.499000 - Le train 3 est arrivé à la gare : E

Le temps moyen du train 3 est de : 2.594000 secondes

2.000000 - Train 3 : E --> A
2.626000 - Le train 1 est arrivé à la gare : B

Le temps moyen du train 1 est de : 2.638000 secondes

2.000000 - Train 1 : B --> A
2.692000 - Le train 2 est arrivé à la gare : D

2.000000 - Train 2 : D --> C
2.789000 - Le train 3 est arrivé à la gare : A

2.811000 - Train 3 en approche en sens inverse A --> B
2.849000 - Le train 2 est arrivé à la gare : C

2.000000 - Train 2 : C --> B
2.944000 - Le train 1 est arrivé à la gare : A

2.000000 - Train 1 : A --> B

3.007000 - Le train 2 est arrivé à la gare : B

Le temps moyen du train 2 est de : 3.074000 secondes

3.091000 - Train 2 en approche en sens inverse B --> A
3.116000 - Le train 3 est arrivé à la gare : B

3.000000 - Train 3 : B --> D
3.147000 - Le train 3 est arrivé à la gare : D

3.000000 - Train 3 : D --> C
3.178000 - Le train 3 est arrivé à la gare : C

3.000000 - Train 3 : C --> E
3.228000 - Le train 2 est arrivé à la gare : A

3.289000 - Train 2 en approche en sens inverse B --> A
3.316000 - Le train 1 est arrivé à la gare : B

3.000000 - Train 1 : B --> C
3.357000 - Le train 3 est arrivé à la gare : E

3.000000 - Train 3 : E --> A
3.411000 - Le train 2 est arrivé à la gare : A

3.000000 - Train 2 : B --> A
3.495000 - Le train 2 est arrivé à la gare : A

3.000000 - Train 2 : A --> B
3.548000 - Le train 2 est arrivé à la gare : B

3.000000 - Train 2 : B --> D
3.603000 - Le train 1 est arrivé à la gare : C

3.000000 - Train 1 : C --> B
3.709000 - Le train 3 est arrivé à la gare : A

3.000000 - Train 3 : A --> B
3.763000 - Le train 2 est arrivé à la gare : D

3.000000 - Train 2 : D --> C
3.876000 - Le train 1 est arrivé à la gare : B

3.939000 -
Train 1 en approche en sens inverse B --> A
3.970000 - Le train 3 est arrivé à la gare : B

3.000000 - Train 3 : B --> D
4.007000 - Le train 2 est arrivé à la gare : C

4.000000 - Train 2 : C --> B

4.062000 - Le train 3 est arrivé à la gare : D

4.000000 - Train 3 : D --> C
4.157000 - Le train 1 est arrivé à la gare : A

4.000000 - Train 1 : B --> A
4.181000 - Le train 1 est arrivé à la gare : A

4.000000 - Train 1 : A --> B
4.220000 - Le train 3 est arrivé à la gare : C

4.000000 - Train 3 : C --> E
4.263000 - Le train 1 est arrivé à la gare : B

4.289000 - Train 1 en approche en sens inverse B --> C
4.309000 - Le train 1 est arrivé à la gare : C

4.000000 - Train 1 : C --> B
4.360000 - Le train 2 est arrivé à la gare : B

4.000000 - Train 2 : B --> A
4.458000 - Le train 1 est arrivé à la gare : B

4.000000 - Train 1 : B --> A
4.531000 - Le train 3 est arrivé à la gare : E

4.582000 - Le train 2 est arrivé à la gare : A

4.645000 - Train 2 en approche en sens inverse A --> B
4.000000 - Train 3 : E --> A
4.687000 - Le train 1 est arrivé à la gare : A

4.000000 - Train 1 : A --> B
4.721000 - Le train 3 est arrivé à la gare : A

4.000000 - Train 3 : A --> B
4.741000 - Le train 3 est arrivé à la gare : B

4.000000 - Train 3 : B --> D
4.764000 - Le train 1 est arrivé à la gare : B

4.000000 - Train 1 : B --> C
4.808000 - Le train 2 est arrivé à la gare : B

4.000000 - Train 2 : A --> B
4.914000 - Le train 2 est arrivé à la gare : B

4.000000 - Train 2 : B --> D
…

Partie 4:

La partie 4 est composée d'une file de message. La file est nommée en QUEUE et persistante. Elle peut être partagée entre 2 processus. On retrouve les fonction open/close/unlink comme pour un sémaphore nommé.
Nous avons créé deux fichiers reader.c et writer.c
Dans le fichier writer.c:

- on définie la taille de buf et la taille de message
- on déclare des éléments de debut_train, fin_train
- on déclare trois éléments, il permet de limiter le calcul de la moyenne à 3 fois.
- on déclare a, b, et c pour les ordre des trajets pour chaque train.
- on déclare stream pour stocker les messages, et le temps.
- on crée une fonction char *substring(size_t start, size_t stop, const char *src, char *dst, size_t size) pour récupérer le caractère du début et de la fin de la chaine de caractere (ex: A --> B , donne A et B).
- on crée trois fonctions TrainUn(),TrainDeux(),TrainTrois() pour comparer les trois trains et afficher des trajets en utilisant sprintf(). Si un train va de A vers B, alors aucun train ne peut aller de B vers A. Par contre, si un train va de A vers B, alors d'autres trains peuvent aller de A vers B.
- dans la fonction main(), nous avons créé un thread pour représenter trois trains, on choisi un train en utilisant rand(). On utilise mq_send pour envoyer un message dans la file.

Cela donne un résultat de fonctionnement avec le fichier writer.c :

```
$ gcc -Wall writer.c -o writer -lrt
$./writer
Queue "/QUEUE":
   - stores at most 10 messages
   -     large at most 8192 bytes each
   - currently holds 5 messages
   -     with 3 trains
train:2
> msg.len = 0
temps:1.000000
Train 2 : A --> B
Le train 2 est arrivé à la gare : B

train:1
> msg.len = 1
temps:2.000000
Train 1 : A --> B
Le train 1 est arrivé à la gare : B

train:3
> msg.len = 2
temps:2.000000
Train 3 : A --> B
Le train 3 est arrivé à la gare : B

train:2
> msg.len = 3
temps:2.000000
Train 2 : B --> D
Le train 2 est arrivé à la gare : D

train:1
```

> msg.len = 4
temps:2.000000
Train 1 : B --> C
Le train 1 est arrivé à la gare : C

train:3
> msg.len = 5
temps:2.000000
Train 3 : B --> D
Le train 3 est arrivé à la gare : D

train:3
> msg.len = 6
temps:3.000000
Train 3 : D --> C
Le train 3 est arrivé à la gare : C

train:3
> msg.len = 7
temps:3.000000
Train 3 : C --> E
Le train 3 est arrivé à la gare : E

train:1
> msg.len = 8
temps:3.000000
Train 1 : C --> B
Le train 1 est arrivé à la gare : B

train:2
> msg.len = 9
temps:3.000000
Train 2 : D --> C
Le train 2 est arrivé à la gare : C

train:3
> msg.len = 10
temps:3.000000
Train 3 : E --> A
Le train 3 est arrivé à la gare : A
Le temps moyen du train 3 est de : 4.074000 secondes

train:1
> msg.len = 11
temps:4.000000
Train 1 : B --> A
Le train 1 est arrivé à la gare : A
Le temps moyen du train1 est de : 4.479000 secondes

train:3
> msg.len = 12
Train 3 en approche en sens inverse A --> B

train:2
> msg.len = 13
temps:4.000000
Train 2 : C --> B
Le train 2 est arrivé à la gare : B

train:2
> msg.len = 14
Train 2 en approche en sens inverse B --> A

train:3
> msg.len = 15
temps:4.000000
Train 3 : B --> D
Le train 3 est arrivé à la gare : D

train:1
> msg.len = 16
Train 1 en approche en sens inverse A --> B
……

Dans le fichier reader.c, il permet respectivement de recevoir un message dans la file en utilisant mq_receive. Il affiche les résultats qui sont les même que dans writer.c

Cela donne un résultat de fonctionnement avec le fichier reader.c :

$ gcc -Wall reader.c -o reader -lrt
$./reader
Queue "/QUEUE":
- stores at most 10 messages
- large at most 8192 bytes each
- currently holds 5 messages
- with 3 trains
train:2
> msg.len = 0
temps:1.000000
Train 2 : A --> B
Le train 2 est arrivé à la gare : B

train:1
> msg.len = 1
temps:2.000000
Train 1 : A --> B
Le train 1 est arrivé à la gare : B

train:3
> msg.len = 2
temps:2.000000
Train 3 : A --> B
Le train 3 est arrivé à la gare : B
…..
On affiche les même résultats entre le fichier writer.c et reader.c.

Conclusion:

On a réalisé les quatre méthodes utilisent la même idée, c'est-à-dire en voulant empêcher les trains de circuler en sens inverse, en utilisant les conditions pour verrouiller et déverrouiller.

On compare les quatres parties des programmes, la partie 4 utilise plus de temps pour finir un tour complet, parce que nous avons utilisé un seul thread pour faire les fils de messages. Les autres parties ont un temps de trajet moyen équivalent.