Projet Lo41

Li Dachen

INFO 02

Professeur:

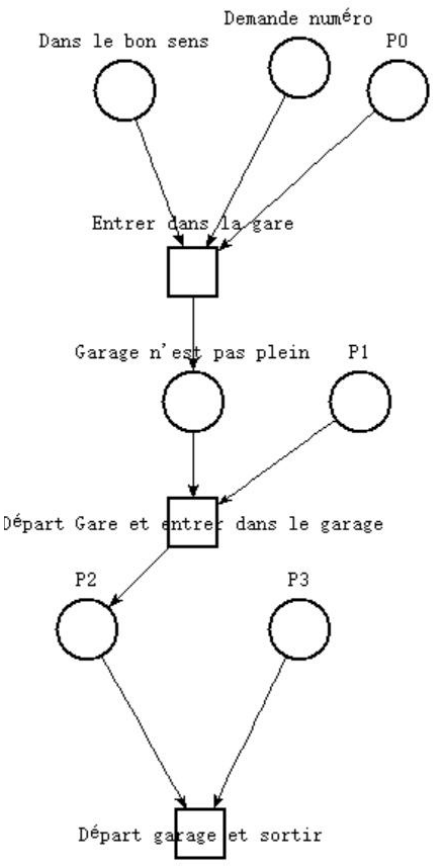
Philippe DESCAMPS **16/06/2016**

1. Considérations générales

Les trains sont considérés comme les threads dans un système d'exploitation et les voies sont considérés comme les ressources dans ce système. Chaque train va utiliser les voies par un séquence prédéfini. Donc on a beoins des variables et les outils dans Linux (mutex, monitor, sémaphore, etc) pour gérer les déroulements des trains.

2. Diagramme de Pétri

TGV de sens Ouest ---> East



Avant le TGV entrer dans la voie C, il faut vérifier s'il existe ou pas les TGV dans l'autre sens. Après il va demander un numéro unique pour l'identifier. Finalement il va demander le signal envoyé par la poste d'aiguillage P0 pour entrer dans la gare. Dans la gare ce TGV va d'abord dispose une durée d'attente en gare. Après cette durée il doit vérifier que le garage de TGV n'est pas plein (si le garage est plein, il doit attendre un signal envoyé par le garage). Il doit aussi demander le signal envoyé par P1 qui veut dire que la voie entre la gare et P1 est libre. Une fois ce TGV est entré dans le garage, Il demande les signaux envoyé par P2 et P3 pour savoir si le tunnel est vide ou pas. S'il a reçu les signaux envoyé par P2 et P3, il peux finalement sortir.

GL de sens Ouest ---> East

Les grande lignes de sens ouest-est suivent les même séquences comme les TGV de sens ouest-est. Mais comme il est moins prioritaire que le TGV, il doit vérifer les nombres de TGV dans deux cas:

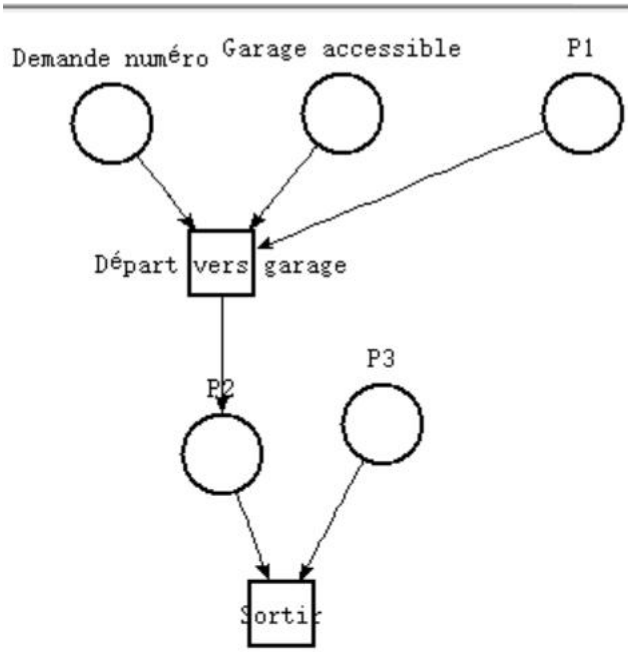
1. Avant son départ:
    a) Il n'existe pas des TGV de sens ouest-est qui est en train d'attendre d'entrer

dans la gare.

b) Il n'existe pas des TGV de sens est-ouest qui veut partir le garage.

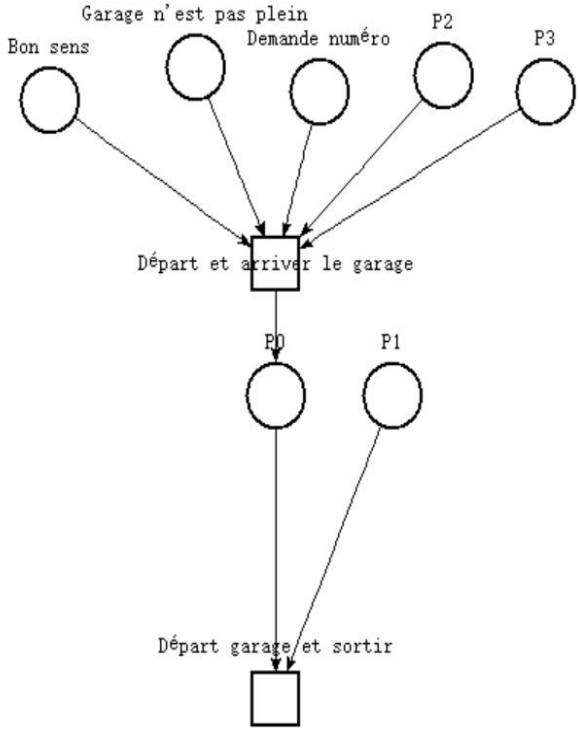
2. Quand il veut partir le garage:
   a) Il n'existe pas des TGV de sens est-ouest qui est en train d'attendre d'entrer
   b) Il n'existe pas des TGV de sens ouest-est qui veut partir le garage.

M de sens Ouest ---> East



Par rapport aux TGV ou GL de sens ouest-est, les trains de marchandises n'ont pas besoin d'attendre le signal de P0 et pour le sens non plus comme pour les trains de marchandises ils ont deux garages de sens différents. Par contre comme il est le moindre prioritaire, quand il veut entrer dans la gare et aussi quand il veut sortir le garage, il doit attendre pas seulement les TGV(comme les GL) mais aussi les GL de la même manière comme les GL attendre les TGV.

TGV de sens East ---> Ouest



Avant le TGV entrer dans la gare, il faut vérifier qu'il n'existe pas les TGV dans l'autre sens et aussi que le garage de TGV n'est pas plein. Après il va demander un numéro unique pour l'identifier. Finalement il va demander le signal envoyé par la poste d'aiguillage P2 et P3 pour passer le tunnel et entrer dans le garage. Dès qu'il entre dans le garage, il va demander les permissions de P0 et P1 pour finalement sortir la gare.

GL et M de sens East ---> Ouest

Ils suivent les même séquences comme les TGV de sens east-ouest sauf que pour les trains de marchandise ils n'ont pas besoin le siganl de P0 comme ils ne vont pas passer la poste P0 et la vérification de sens non plus. Concernant le prioritaire, il existe aussi quatres cas pour les GL(attendre les TGV) et huit cas pour les M(attendre les TGV et aussi les GL).

1. Avant son départ:
   a) Il n'existe pas des TGV de sens est-ouest qui est en train d'attendre d'entrer dans la gare.
   b) Il n'existe pas des TGV de sens ouest-est qui veut partir le garage.
2. Quand il veut partir le garage:
   a) Il n'existe pas des TGV de sens ouest-est qui est en train d'attendre d'entrer
   b) Il n'existe pas des TGV de sens est-ouest qui veut partir le garage.

3. Analyse fonctionnelle et algorithmique

Choix des variables

Mutex

Utilisé pour:

1. Poste d'aiguillage(P0, P1, P2, P3)
   Ex: *pthread_mutex_t P0=PTHREAD_MUTEX_INITIALIZER;*
2. Protecter les variables globales qui sont utilisées par tous les threads.
   Ex: *pthread_mutex_t sensTGV_mut=PTHREAD_MUTEX_INITIALIZER;*

Comme dans un moment il peut exister seulement un train sur une partie de la voie, il faut que le thread qui est entrain de rouler sur cette partie de la voie vérouiller les postes d'aiguillage concernants enfin d'éviter les collisions. Quand le train est enfin partir, il doit déverrouiller les postes d'aiguillage utilisées.

Monitor

Utilisé pour:

1. Informer les TGV et GL mise en attente quand il n'existe plus le même type de train dan le sens contraire
   Ex: *pthread_cond_t TGV_oe_cond=PTHREAD_COND_INITIALIZER;*
   pthread_cond_t TGV_eo_cond=PTHREAD_COND_INITIALIZER;
2. Informer les trains moins prioritaire (GL ou M) de partir quand il n'existe plus les contraintes crées par les trains qui sont plus prioritaires.

Ex: *pthread_cond_t entrer_TGV_oe_cond=PTHREAD_COND_INITIALIZER;*
pthread_cond_t garage_TGV_oe_cond=PTHREAD_COND_INITIALIZER;
pthread_cond_t entrer_GL_eo_cond=PTHREAD_COND_INITIALIZER;
pthread_cond_t garage_GL_eo_cond=PTHREAD_COND_INITIALIZER;

Les monitors sont utilisé quand il existe une variable qui présente une condition. Par exemple pour présenter le sens des TGV dans la gare dans un moment, j'ai utilisé une variable *int sensTGV*(supérieur à 0 si les TGV dans la gare sont de sens ouest-est, inférieur à 0 si les TGV dans la gare sont de sens est-ouest, égale à 0 si il n'y pas de TGV dans ce moment). Si dans un moment un TGV de sens ouest-est va entrer dans la gare et la variable *sensTGV<0*, il va attendre le signal(*pthread_cond_t*) *TGV_oe_cond* qui est envoyé par le dernier TGV de sens est-ouest dans la gare quand ce TGV est parti de la gare.

Sémaphore

Utilisé pour:
Gérer les garages des différent type de train
Ex: sem_t sem_garageTGV;
sem_t sem_garageM_oe;
Les sémaphores sont utilisés quand il existe une variable qui présente une condition comme le cas de l'utilisation d'un monitor sauf que cette fois cette variable a une valeur maximum. Par example, le garage de chaque genre de train a une capacité(c'est différent de l'example dans la partie de monitor car dans un moment le nombre de TGV dans l'autre sens n'a pas de contrainte). Donc si il n'existe plus de place dans le garage, le nouveau train ne peut pas partir et il va attendre le départ d'un train qui est du même type.

La vie d'un thread(train)

Pour chaque type de train, il existe une variable *int nb_TYPE_SENS* pour dire le nombre de train existe dehors la gare. Quand cette variable est supérieur à 0, le processus va créer un thread qui utilise la fonction pour ce genre de train et diminuer le nombre de train par 1.

Ici je voudrais présenter une partie de code pour la création d'un thread de GL de sens ouest-east car cette partie contient le plus part d'information dans la vie d'un thread.

```
void *GL_oe( )
{
```

/* La vérification de sens. Si il existe déja les GL dans l'autre sens(*sensGL<0*), ce thread va attendre le signal *GL_oe_cond* et déverrouiller le mutex *sensGL_mut* pour que les autres threads peuvent modifier la valeur de *sensGL*. Quand il a reçu le signal

GL_oe_cond, il va incrémenter la valeur sens GL pour dire que l'on a ajouté un GL de sens ouest-east(pour les GL de sens east-oueast ici on droit décrémenter la valeur de sensGl, et quand il part, il faut incrémenter cette valeur) */

```
pthread_mutex_lock(&sensGL_mut);
if (sensGL<0) pthread_cond_wait(&GL_oe_cond,&sensGL_mut);
sensGL++;
pthread_mutex_unlock(&sensGL_mut);
```

/*Avant son départ, il va informer les M de sens ouest-east qu'ils doivent attendre. Dans la fonction de la création de M sens oe, il y a donc une monitoi qui va voir si *nb_GL_oe_entrer >0*, si oui, le M doit attendre le signal envoyé par le dernier GL de sens ouest-east. */

```
pthread_mutex_lock(&entrer_GL_oe_mut);
nb_GL_oe_entrer++;
pthread_mutex_unlock(&entrer_GL_oe_mut);
//Ici j'ai utilisé une variable locale(num) pour présenter son idendité.
pthread_mutex_lock(&num_mut);
int num=numtrain;
numtrain++;
pthread_mutex_unlock(&num_mut);
```

/* Vérification de TGV_oe_entrer et TGV_eo_garage. Si il existe au moins une variable qui n'est pas 0, il doit attendre le signal. */

```
pthread_mutex_lock(&entrer_TGV_oe_mut);
if(nb_TGV_oe_entrer!=0)pthread_cond_wait(&entrer_TGV_oe_cond,&entrer_TGV_oe_mut);
pthread_mutex_unlock(&entrer_TGV_oe_mut);

pthread_mutex_lock(&garage_TGV_eo_mut);
if(nb_TGV_eo_garage!=0)pthread_cond_wait(&garage_TGV_eo_cond,&garage_TGV_eo_mut);
pthread_mutex_unlock(&garage_TGV_eo_mut);
```

/* Quand l'environement est bon, le GL va demander la permission de P0 pour partir. Ces demandes doivent toujours avant l'affichage des phrases.*/

```
pthread_mutex_lock(&P0);
printf ("GL-->%d attend\n",num);
sleep(1);
// La vérification du garage de GL et demande la permission de P1
sem_wait(&sem_garageGL);
pthread_mutex_lock(&P1);
printf("GL-->%d départ Gare\n",num);
```

/* Juste après l'affichage(qui veut dire que ce GL est bien départ la gare) ce GL peut déverrouiller le P0.*/

```
pthread_mutex_unlock(&P0);
sleep(1);
```

/* Avant son arrive dans le garage il faut incrémenter le nombre de train dans le

garage. Si on le fait après l'affichage, les autres GL peuvent être lancé pendant la période entre l'affichage et le vérrouillage du sémaphore même le garage est déja plein.*/

```
pthread_mutex_lock(&garage_GL_oe_mut);
nb_GL_oe_garage++;
pthread_mutex_unlock(&garage_GL_oe_mut);
printf(" %28s%d %s\n","GL-->",num,"arrive Garage");
```

/*Quand le GL est dans le garage, il décrémente le nombre de GL de sens oe qui veut entrer dans la gare. Et si après la décrémentation cette valeur égale à 0, il va envoyer le sémaphore *entrer_GL_oe_cond* à tous les threads qui l'attendent (M de sens oe qui veulent entrer et M de sens eo qui veulent partir le garage)*/

```
pthread_mutex_lock(&entrer_GL_oe_mut);
nb_GL_oe_entrer--;
if(nb_GL_oe_entrer==0)pthread_cond_broadcast(&entrer_GL_oe_cond);
pthread_mutex_unlock(&entrer_GL_oe_mut);
pthread_mutex_unlock(&P1);
```

/*Ici c'est un test pour vérifier que le sémaphore de garageGL fonctionne.

On peut initialiser seulement les GL de sens ouest-east, et comme avant son départ du garage il doit attendre 20 seconde, les autres GL peuvent aussi arriver dans le garage et donc on peux voir le nombre de GL maximum dans le garage. */

```
//sleep(20); test sem
```

// La même manière comme en début la vérification pour entrer dans la gare.

```
pthread_mutex_lock(&garage_TGV_oe_mut);
if(nb_TGV_oe_garage!=0)pthread_cond_wait(&garage_TGV_oe_cond,&garage_
TGV_oe_mut);
pthread_mutex_unlock(&garage_TGV_oe_mut);

pthread_mutex_lock(&entrer_TGV_eo_mut);
if(nb_TGV_eo_entrer!=0)pthread_cond_wait(&entrer_TGV_eo_cond,&entrer_TG
V_eo_mut);
pthread_mutex_unlock(&entrer_TGV_eo_mut);
```

/* C'est toujours après la vérification de la priorité qu'il peut demande les permissions des postes d'aiguillage. Sinon il y aura un interblocage. Par exemple un GL d'abord reçu les permissions de P2 et P3 et attendre les TGV partir, et dans le même temps un TGV est dans le garage et attend P2 et P3. Dans ce cas là tous les trains ne peuvent pas rouler. */

```
pthread_mutex_lock(&P2);
pthread_mutex_lock(&P3);
printf(" %28s%d %s\n","GL-->",num,"départ Garage");
```

// Informer les M de partir s'il est le dernier train dans le garage GL.

```
pthread_mutex_lock(&garage_GL_oe_mut);
nb_GL_oe_garage--;
if(nb_GL_oe_garage==0)pthread_cond_broadcast(&garage_GL_oe_cond);
pthread_mutex_unlock(&garage_GL_oe_mut);
```

```
    // Informer les GL de partir comme il y a au moins une place dans le garage
    sem_post(&sem_garageGL);

    printf(" %53s%d %s\n","GL-->",num,"dans Tunnel");
    sleep(1);
    printf(" %83s%d %s\n","GL-->",num,"sort");
    sleep(1);
    pthread_mutex_unlock(&P2);
    pthread_mutex_unlock(&P3);
    /* Informer les GL dans l'autre sens de partir s'il est le dernier GL de sens ouest-east */
    pthread_mutex_lock(&sensGL_mut);
    sensGL--;
    if (sensGL==0) pthread_cond_broadcast(&GL_eo_cond);

    pthread_mutex_unlock(&sensGL_mut);

    pthread_exit(NULL);
}
```