Systèmes temps réel TP1

Adrien FABRE & Thomas CHEKROUN

20/02/2015

Exercices 1 et 2

Les implémentations respectives de ces exercices sont dans les fichiers exercice1.c et exercice2.c.

Le but de ces exercices étaient une première application basique des notions vues en cours.

Exercice 3

Question 3

Les implémentations sont les suivantes:

Pour le thread écriture :

Pour i allant de 0 à nombre d'écritures souhaitées

P(libres)

lock(mutex)

écrire dans buffer au niveau de la tête d'écriture

déplacer la tête d'écriture à la case suivante

unlock(mutex)

V(occupées)

FinPour

On effectue un P(libres) pour ne commencer l'écriture que si une case est disponible pour être remplie. Une fois l'écriture effectuée on utilise V(occupées) pour indiquer au lecteur qu'il y a une case à lire.

L'exclusion mutuelle (mutex) permet de protéger la lecture et le déplacement de la tête de lecture dans le cas où plusieurs écrivains sont présents.

Pour le thread lecture:

Pour i allant de 0 à nombre de lectures souhaitées

P(occupées)

lire dans le buffer au niveau de la tête de lecture

déplacer la tête de lecture à la case suivante

V(libres)

FinPour

Dans le processus de lecture, on effectue un P(occuppées) pour ne commencer la lecture que si une case contient une donnée à lire. Inversement une fois la case lu on signale à l'écrivain qu'une nouvelle case est disponible à l'écriture.

Question 4

On a un buffer de N cases. Le lecteur peut lire les cases écrites, l'écrivain peut écrire dans les cases vides. Les deux threads peuvent s'exécuter en même temps. On met un sémaphore sur les cases occupées et un sur les cases libres. Le sémaphore occupé est initialisé à 0 pour représenter le fait qu'au début de l'exécution toutes les cases sont vides.Le sémaphore vide est initialisé à N pour représenter le fait qu'au début de l'exécution toutes les cases sont disponibles à la lecture.

De plus pour modéliser notre buffer nous avons besoin d'une tête de lecture et d'une tête d'écriture. Ces têtes seront des pointeurs indiquant sur quelles cases les processus peuvent effectuer leurs actions. Ces pointeurs sont tous les deux initialisés à 0 car la lecture et l'écriture se font en partant de la première case du buffer.

Question 5

cf. exercice3.c pour le problème avec 1 lecteur et 1 écrivain et

cf. exercice3.1.c pour le problème avec 1 lecteur et 3 écrivains.

Exercice 4

Question 6

Le problème de type d'exclusion mutuelle est lié à la piste d'atterrissage/décollage. En effet les threads permettant d'atterrir et de décoller ne doivent pas pouvoir utiliser cette piste en même temps.

Le problème de type producteur/consommateur est lié aux zones d'attentes:

Lorsque ces zones sont pleines il faut attendre qu'un avion en sorte avant de pouvoir en rajouter un nouveau.

Nous aurons besoin de 1 exclusion mutuelle (mutex), initialisée à 1. Elle permettra aux threads d'atterrissage et de décollage de ne pas utiliser la même piste.

Il faudra également 4 sémaphores:

Il y en aura 2 par zones d'attentes en se basant sur les mêmes que le buffer de l'exercice précédent

Air: occupées initialisé à 0

libre initialisé à N

Sol: occupées initialisé à 0

libre initialisé à M

Question 7

Les quatre processus existants sont les suivants:

AmenerAvion : correspond au remplissage de la file d'attente des avions dans l'air

SortirAvion : correspond au remplissage de la file d'attente des avions au sol

Atterrissage : correspond à la libération de la file d'attente des avions dans l'air

Décollage : correspond à la libération de la file d'attente des avions au sol

AmenerAvion:

```
Pour i de 0 au nombre d'avions total en l'air
   P(libresAir)
   lock(mutAir)
   Air[AirWritingHead] = nouvelAvion
   Décalage de AirWritingHead
   unlock(mutAir)
   V(occuppeesAir)
FinPour
```

SortirAvion:

```
Pour i de 0 au nombre d'avions total au sol
   P(libresSol)
   lock(mutSol)
   Sol[SolWritingHead] = nouvelAvion
   Décalage de SolWritingHead
   unlock(mutSol)
   V(occuppeesSol)
FinPour
```

Atterrissage:

Pour i de 0 au nombre d'avions total en l'air

P(occuppeesAir)

lock(pisteAtterrissage)

Décalage de AirReadingHead

unlock(pisteAtterrissage)

V(libresAir)

FinPour

Décollage:

Pour i de 0 au nombre d'avions total au sol

P(occuppeesSol)

lock(pisteAtterrissage)

Décalage de SolReadingHead

unlock(pisteAtterrissage)

V(libresSol)

FinPour

Question 8

L'ordre des prises des sémaphores est important. En effet en inversant les instructions P et V, les threads risques d'écrire dans les files d'attente sur des emplacements qui n'avaient pas encore été traités ce qui engendre un comportement imprévisible de l'application.

De plus inclure les appels P() et V() dans les zones d'exclusion mutuelles pourrait conduire à un blocage de l'application: les threads ne pouvant plus accéder à la piste, dans certains cas ils ne pourraient plus faire évoluer leurs files d'attentes.

Question 10

L'implémentation se trouve dans le fichier exercice4.c