Licence 3 Informatique

Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 09

Programmation Système
Travaux pratiques n°2 & 3– Synchronisation par sémaphores

Exercice 1 – Alternance d'affichage

Préparation [TP2] : Synchronisation théorique, comme en TD (P/V).
Rendu Moodle : Code associé à chaque version du programme demandé.
Date limite : Fin de la 2e séance de TP.
Code disponible sous Moodle : tp23_exo1_base.c

Compléter la base fournie (fonctions P, V, demanderAcces et libererAcces) afin de synchroniser l'affichage des N threads lancés de manière à ce qu'ils affichent leur message à tour de rôle à l'écran (même principe que l'exercice vu en TD).

Code disponible sous Moodle :

- tp23_exo1_base.c qui fournit le squelette de code commun aux 2 versions demandées.

❖ **Version 1** : La synchronisation sera réalisée à l'aide de verrous d'exclusion mutuelle Posix (**pthread_mutex_t**).

- Le fichier rendu aura pour nom : **VotreNomPrenom**_tp23_exo1-v1.c

❖ **Version 2** : La synchronisation sera réalisée à l'aide de sémaphores Posix (**sem_t**).

- Le fichier rendu aura pour nom : **VotreNomPrenom**_tp23_exo1-v2.c

Exemple d'exécution :

%./tp23_exo1-v1 4

[Ici, chacun des 4 afficheurs affiche 2 messages de 2 lignes]

Afficheur 0 (140592487073536), j'affiche ligne 1/2 du message 1/2
Afficheur 0 (140592487073536), j'affiche ligne 2/2 du message 1/2
Afficheur 1 (140592478619392), j'affiche ligne 1/2 du message 1/2
Afficheur 1 (140592478619392), j'affiche ligne 2/2 du message 1/2
Afficheur 2 (140592470165248), j'affiche ligne 1/2 du message 1/2
Afficheur 2 (140592470165248), j'affiche ligne 2/2 du message 1/2
Afficheur 3 (140592461711104), j'affiche ligne 1/2 du message 1/Z
Afficheur 3 (140592461711104), j'affiche ligne 2/2 du message 1/2
Afficheur 0 (140592487073536), j'affiche ligne 1/2 du message 2/2
Afficheur 0 (140592487073536), j'affiche ligne 2/2 du message 2/2
Afficheur 0 (140592487073536), je me termine
Afficheur 1 (140592478619392), j'affiche ligne 1/2 du message 2/2
Afficheur 1 (140592478619392), j'affiche ligne 2/2 du message 2/2
Afficheur 1 (140592478619392), je me termine
Afficheur 2 (140592470165248), j'affiche ligne 1/2 du message 2/2
Afficheur 2 (140592470165248), j'affiche ligne 2/2 du message 2/2
Afficheur 2 (140592470165248), je me termine
Afficheur 3 (140592461711104), j'affiche ligne 1/2 du message 2/2
Afficheur 3 (140592461711104), j'affiche ligne 2/2 du message 2/2

Afficheur 3 (140592461711104), je me termine

Fin de l'execution du thread principal

Exercice 2 – Réaliser la synchronisation demandée (Session 2 – Juin 2017)

Préparation [TP2] : Synchronisation théorique, comme en TD (P/V).
Rendu Moodle : Code associé au programme demandé.
Date limite : 03/11/2019 minuit.
Code disponible sous Moodle : tp23_exo2_base.c

Code disponible sous Moodle :

- tp23_exo2_base.c qui crée les threads dont il faut synchroniser l'exécution.

On considère les trois threads cycliques T1, T2 et T3 suivants :

Thread T1 {
Boucler {
Afficher 'A' ;
Afficher 'B' ;
}
}
	Thread T2 {
	Boucler {
	Afficher 'C' ;
	}
	}
		Thread T3 {
		Boucler {
		Afficher 'D' ;
		}
		}

❖ Synchroniser ces threads de manière à restreindre les affichages possibles (à chaque « cycle ») à ACDB **ou bien** ADCB.

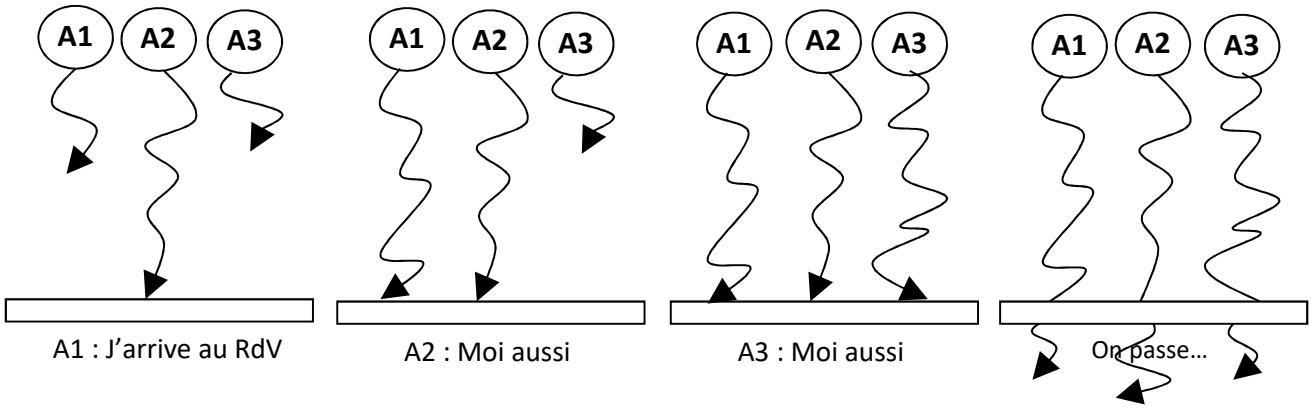
- Le fichier rendu aura pour nom : **VotreNomPrenom**_tp23_exo2.c

Exercice 3 – Rendez-vous entre threads

Préparation [TP3] : Synchronisation théorique, comme en TD (P/V).
Rendu Moodle : Code associé au programme demandé.
Date limite : Fin de la 3e séance de TP.
Code disponible sous Moodle : tp23_exo3_base.c

On désire réaliser un rendez-vous entre N activités parallèles : une activité arrivant au point de rendez-vous se met en attente s'il existe au moins une autre activité qui n'y est pas arrivé. Toutes les activités bloquées sur cette « barrière » peuvent la franchir lorsque la dernière y est arrivée.

La figure ci-dessous illustre ce comportement pour un rendez-vous à 3 :



Une activité a le comportement suivant :

```
Début
     Je fais un certain traitement ;
     J'arrive au point de rendez-vous
          et j'attends que tous les autres y soient aussi… ;
     …Avant de pouvoir continuer mon traitement ;
Fin
```

Code disponible sous Moodle :

- tp23_exo3_base.c qui fournit un squelette de programme.

❖ Le fichier rendu aura pour nom : **VotreNomPrenom**_tp23_exo3.c

Exemple d'exécution :

%./tp23_exo3 4

[Ici, 4 threads réalisent un rendez-vous]

Thread 0 (140068121478912) : J'effectue un traitement en parallele avec les autres
Thread 0 (140068121478912) : J'arrive au RdV
Thread 1 (140068113024768) : J'effectue un traitement en parallele avec les autres
Thread 1 (140068113024768) : J'arrive au RdV
Thread 2 (140068104570624) : J'effectue un traitement en parallele avec les autres
Thread 2 (140068104570624) : J'arrive au RdV
Thread 2 (140068104570624) : Je passe le point de RdV
Thread 2 (140068104570624) : Je continue un traitement en parallele avec les autres
Thread 0 (140068121478912) : Je passe le point de RdV
Thread 1 (140068113024768) : Je passe le point de RdV

Thread 1 (140068113024768) : Je continue un traitement en parallele avec les autres
Thread 0 (140068121478912) : Je continue un traitement en parallele avec les autres
Fin de l'execution du thread principal

Exercice 4 – Modèle des producteurs-consommateurs

Préparation [TP3] : Synchronisation théorique, comme en TD (P/V).
Rendu Moodle : Code associé au programme demandé.
Date limite : 17/11/2019 minuit.
Code disponible sous Moodle : tp23_exo4_base.c

On s'intéresse ici au modèle producteurs/consommateurs étudié en cours-TD. Le tampon commun peut supporter jusqu'à N messages et est géré circulairement : les dépôts doivent se faire dans l'ordre croissant des indices de cases, de manière circulaire ; les retraits se font dans l'ordre des dépôts, de manière circulaire aussi.

Code disponible sous Moodle :

- tp23_exo4_base.c qui fournit un squelette de programme.

❖ Synchroniser les threads producteurs et consommateurs de l'application pour qu'ils déposent et retirent leurs messages de manière cohérente.

- Le fichier rendu aura pour nom : **VotreNomPrenom**_tp23_exo4.c

Exemples d'exécution :

%./tp23_exo4 3 3 4 4 1

[Ici, 3 producteurs et 3 consommateurs faisant chacun 4 dépôts/retraits dans un buffer de 1 case]

Prod 2 (140388782180096) : Message depose = Bonjour 1 de prod 2
Conso 0 (140388773725952) : Message retire = Bonjour 1 de prod 2
Prod 1 (140388790634240) : Message depose = Bonjour 1 de prod 1
Conso 2 (140388756817664) : Message retire = Bonjour 1 de prod 1
Prod 2 (140388782180096) : Message depose = Bonjour 2 de prod 2
Conso 0 (140388773725952) : Message retire = Bonjour 2 de prod 2
Prod 0 (140388799088384) : Message depose = Bonjour 1 de prod 0
Conso 1 (140388765271808) : Message retire = Bonjour 1 de prod 0
Prod 1 (140388790634240) : Message depose = Bonjour 2 de prod 1
Conso 1 (140388765271808) : Message retire = Bonjour 2 de prod 1
Prod 1 (140388790634240) : Message depose = Bonjour 3 de prod 1
Conso 0 (140388773725952) : Message retire = Bonjour 3 de prod 1
Prod 2 (140388782180096) : Message depose = Bonjour 3 de prod 2
Conso 2 (140388756817664) : Message retire = Bonjour 3 de prod 2
Prod 0 (140388799088384) : Message depose = Bonjour 2 de prod 0
Conso 0 (140388773725952) : Message retire = Bonjour 2 de prod 0
Prod 0 (140388799088384) : Message depose = Bonjour 3 de prod 0
Conso 1 (140388765271808) : Message retire = Bonjour 3 de prod 0
Prod 2 (140388782180096) : Message depose = Bonjour 4 de prod 2
Conso 2 (140388756817664) : Message retire = Bonjour 4 de prod 2
Prod 1 (140388790634240) : Message depose = Bonjour 4 de prod 1

Conso 1 (140388765271808) : Message retire = Bonjour 4 de prod 1
Prod 0 (140388799088384) : Message depose = Bonjour 4 de prod 0
Conso 2 (140388756817664) : Message retire = Bonjour 4 de prod 0

Fin de l'execution du main

%./tp23_exo4 3 3 4 4 2

[Ici, 3 producteurs et 3 consommateurs faisant chacun 4 dépôts/retraits dans un buffer de 2 cases]

Prod 1 (140588504516352) : Message depose = Bonjour 1 de prod 1
Prod 2 (140588496062208) : Message depose = Bonjour 1 de prod 2
Conso 2 (140588470699776) : Message retire = Bonjour 1 de prod 1
Conso 1 (140588479153920) : Message retire = Bonjour 1 de prod 2
Prod 0 (140588512970496) : Message depose = Bonjour 1 de prod 0
Prod 2 (140588496062208) : Message depose = Bonjour 2 de prod 2
Conso 0 (140588487608064) : Message retire = Bonjour 1 de prod 0
Prod 1 (140588504516352) : Message depose = Bonjour 2 de prod 1
Conso 0 (140588487608064) : Message retire = Bonjour 2 de prod 2
Conso 1 (140588479153920) : Message retire = Bonjour 2 de prod 1
Prod 0 (140588512970496) : Message depose = Bonjour 2 de prod 0
Prod 1 (140588504516352) : Message depose = Bonjour 3 de prod 1
Conso 2 (140588470699776) : Message retire = Bonjour 2 de prod 0
Prod 2 (140588496062208) : Message depose = Bonjour 3 de prod 2
Conso 2 (140588470699776) : Message retire = Bonjour 3 de prod 1
Conso 0 (140588487608064) : Message retire = Bonjour 3 de prod 2
Prod 0 (140588512970496) : Message depose = Bonjour 3 de prod 0
Prod 2 (140588496062208) : Message depose = Bonjour 4 de prod 2
Conso 1 (140588479153920) : Message retire = Bonjour 3 de prod 0
Prod 1 (140588504516352) : Message depose = Bonjour 4 de prod 1
Conso 0 (140588487608064) : Message retire = Bonjour 4 de prod 2
Conso 1 (140588479153920) : Message retire = Bonjour 4 de prod 1
Prod 0 (140588512970496) : Message depose = Bonjour 4 de prod 0
Conso 2 (140588470699776) : Message retire = Bonjour 4 de prod 0

Fin de l'execution du main

Exercice 4b – Modèle des producteurs-consommateurs : variantes

Si vous voulez aller plus loin, vous pouvez implanter les versions assurant l'alternance des dépôts ou le retrait à la demande (voir TD).