Université Paul Sabatier
31062 Toulouse Cedex 09

Master 1 Informatique - MCPR

Modèles et Concepts du Parallélisme et de la Répartition
Travaux pratiques n°3

Synchronisation par sémaphores sous Posix (suite)

Exercice 1

On souhaite implanter une solution au problème des producteurs/consommateurs (étudié en cours et en TD) en utilisant les sémaphores et les segments de mémoire partagée Posix.

1. Écrire une commande, **paramétrée** par le nombre de producteurs et le nombre de consommateurs, permettant de simuler ce modèle de synchronisation lorsque les messages échangés via le tampon partagé sont tous du **même type**.
2. Modifier ce programme pour implanter la variante 1 étudiée au TD2, dans laquelle les producteurs déposent des **messages de deux types**.
3. Éventuellement, modifier ce programme pour implanter la variante 2 étudiée au TD2, dans laquelle les consommateurs précisent le type de messages qu'ils veulent retirer du tampon partagé.

Dans les différentes variantes, les messages sont toujours retirés dans l'**ordre des dépôts**.

Exercice 2

On désire gérer la distribution de carburant dans une station comprenant deux pompes.

La première pompe est à paiement automatique, la seconde dispose d'un paiement en caisse.

Dans le premier cas, un client est capable de se servir s'il dispose de l'accès au pistolet de distribution, le fait de reposer le pistolet libère l'accès pour un éventuel client suivant. Pour se servir à la deuxième pompe, un client doit disposer de l'accès au pistolet de distribution mais aussi attendre que le compteur de cette pompe soit remis à zéro par la caisse.

Un client sera simulé par un processus cyclique ayant le comportement suivant :

```
Début cycle
     Accéder à la pompe choisie (1 ou 2) ;
     Faire le plein ;
     Reposer le pistolet de la pompe ;
     Payer en caisse (si la pompe choisie était la 2) ;
Fin cycle
```

La caisse gérant la pompe manuelle est simulée par un processus cyclique au comportement suivant :

```
Début cycle
     Attendre que le client paye ;
     Remettre à zéro le compteur de la pompe ;
Fin cycle
```

1. En considérant les différents types de processus intervenant dans le problème, écrire la solution théorique permettant de les synchroniser selon les directives ci-dessus.

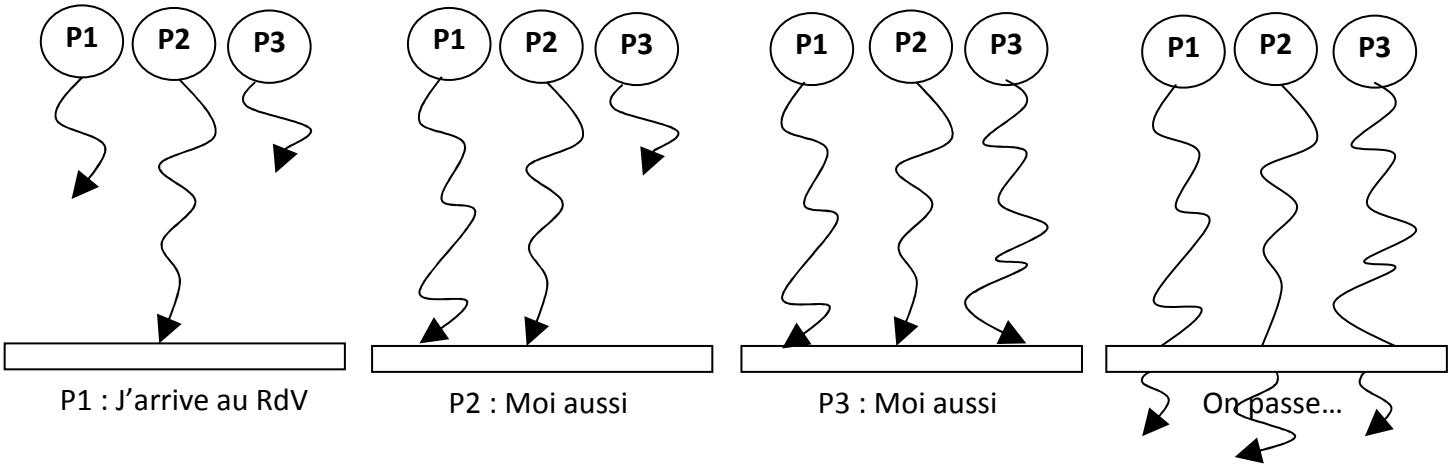
2. Implanter cette solution en écrivant le code relatif :

- À un processus `Caisse` qui gère le paiement et l'accès à la deuxième pompe.
- À un processus `Client` qui souhaite faire son plein, ce processus sera paramétré par le numéro de la pompe choisie.
- Au programme dans lequel interviennent N (valeur variable d'une exécution à l'autre donc paramètre du programme) processus `Client` et un processus `Caisse`.

Pour aller plus loin…

On désire réaliser un rendez-vous entre P processus : un processus arrivant au point de rendez-vous se met en attente s'il existe au moins un autre processus qui n'y est pas arrivé. Tous les processus bloqués sur cette « barrière » peuvent la franchir lorsque le dernier y est arrivé.

La figure ci-dessous illustre ce comportement.



Un processus 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
```

1. Écrire un programme permettant, à l'aide des IPC Posix, de mettre en place ce type de synchronisation entre un nombre **quelconque** de processus (ce nombre pouvant varier d'une exécution à l'autre et est donc paramètre du programme).
2. Si vous ne l'avez pas déjà fait, modifiez-le afin de pouvoir faire s'exécuter chaque processus **dans une fenêtre différente** (i.e. synchroniser des processus sans aucun lien de parenté).