Faculté des Sciences & Techniques — Université de Limoges

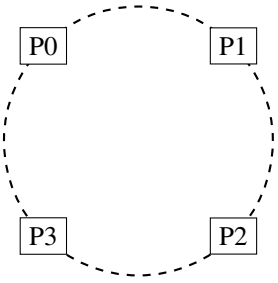
Licence 3ème année

Parallélisme I

Examen – décembre 2020

Durée : 1h30 — Documents autorisés

1 – 13pts On veut créer 4 processus pouvant communiquer entre eux à l'aide d'une boucle constituée de « *tubes* » :

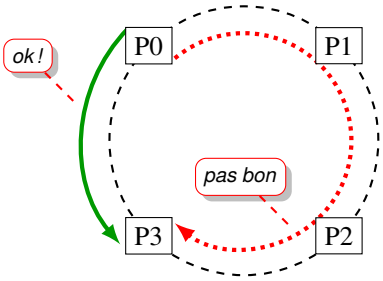


- ☐ Chaque processus est numéroté de 0 à 3 ;
- ☐ P0 communique avec P1, P1 communique avec P2, P2 communique avec P3 et P3 communique avec P0 ;
- ☐ Deux processus qui peuvent communiquer ensemble doivent pouvoir communiquer dans **les deux sens** :
  Exemple :
  Si P0 peut communiquer avec P1 alors ▷ *P0 peut écrire à P1 ;*
  ▷ *P1 peut écrire à P0.*

a. De combien de tubes va-t-on avoir besoin ? Quand doit-on les créer par rapport aux *forks* ? *(1pt)*

b. Écrire le programme réalisant : *(4pts)*
- ◇ la **création des 4 processus** avec chacun son identifiant entre 0 et 3 ;
- ◇ la **mise en place des différents tubes** nécessaires à leur communication dans les deux sens ;

c. On veut réaliser une communication « *optimisée* » : *(4pts)*



Si le processus P_i veut communiquer avec le processus P_j alors il doit choisir le sens de communication passant par le **plus petit nombre** de processus.

Exemple : P0 veut communiquer avec P3 :
- ◇ *communication directe de P0 vers P3 ⟹ok !*
- ◇ *communication de P0 vers P1, puis vers P2 et enfin P3 ⟹pas bon...*

Écrire les instructions à ajouter à chaque processus permettant de :
- ▷ **décider** dans quel sens envoyer le message ;
- ▷ **relayer** les messages qui ne sont pas à notre destination.

Un message a la forme suivante :

```
struct message
{
     int id_processus;
     int valeur;
};
```

Pour illustrer le fonctionnement de vos instructions vous pourrez envoyer un message de P0 vers P3 où P3 affiche la valeur reçue.

d. On voudrait utiliser de la **mémoire partagée** pour échanger entre les différents processus. *(4pts)*

Expliquez :
- ◇ les différences entre l'utilisation de la **mémoire partagée** et l'utilisation de **tubes** ;
- ◇ comment va-t-on résoudre les problèmes d'**accès concurrents**.

Attention *: Vous n'avez pas à écrire de code pour répondre à cette question.*

2 – Soient les 4 threads suivantes, s'exécutant chacune une seule fois :

5pts

```
thread_A {
   P(s2);
   x = x + y;
   V(s2);
}
```

```
thread_B {
   P(s2);
   y = x * y;
   V(s2);
}
```

```
thread_C {
   P(s1);
   y = y * 3;
   V(s3);
}
```

```
thread_D {
   P(s1);
   P(s3);
   x = x + 2;
   V(s2);
}
```

Elles utilisent pour leur synchronisation trois sémaphores `s1`, `s2` et `s3`.

a. Quelles sont les différentes valeurs possibles des variables `x` et `y` à la fin de l'exécution complète des quatres *threads* avec les valeurs d'initialisation suivantes : *(3pts)*

Sémaphore	s1	s2	s3
valeur initiale	2	0	0

variable	x	y
valeur initiale	0	1

b. Qu'est-ce que cela change avec les valeurs d'initialisation suivantes : *(2pts)*

Sémaphore	s1	s2	s3
valeur initiale	1	0	1

variable	x	y
valeur initiale	0	1

3 – Quelles sont les **différences** entre « *signaux* » et « *sémaphores* » ?

2pts