

Epreuve partielle
(Systèmes d'exploitation- M1: IV)

Le 15/11/2023 - Année 22/23

Nom		Prénom		Matricule	
-----	--	--------	--	-----------	--

Exercice 1: (10 pts= 2*5)

- a- Pourquoi la structure Fork/Join est plus expressive que Parbegin Parend ?
- b- Donner le diagramme de temps d'exécution du programme suivant :
Debut Parbegin T1 ; Debut T2 ; T3 Fin Parend ; Parbegin T4 ; T5 Parend ; T6 Fin.
- Les temps d'exécutions des tâches en unités de temps sont : T1 : 2 ; T2 : 1 ; T3 : 3 ;
T4 : 3 ; T5 : 1 ; T6 : 1.
- c- Comment est mise en œuvre l'indivisibilité de l'instruction TAS sur un multiprocesseur?
- d- Comment on a étendu les événements pour surmonter la rigidité de leurs utilisations ?
- e- Que signifie l'expression "*sans priorité explicite*" dans le modèle des Lecteurs/Rédacteurs ?

Exercice 2 : (5 pts)

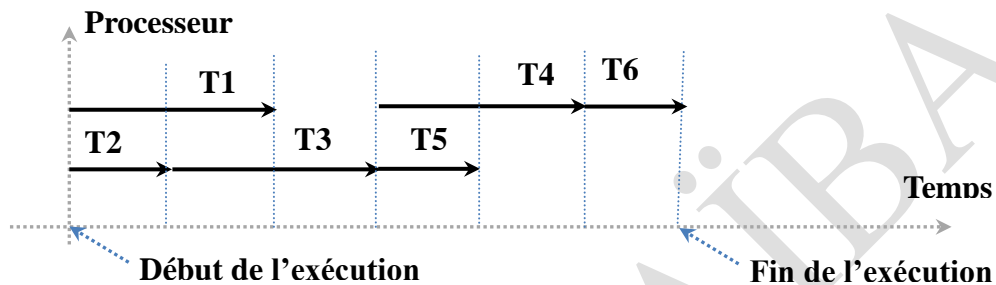
On suppose un système constitué de plusieurs processus et uniquement de trois événements mémorisés $e1$, $e2$ et $e3$. On considère les deux primitives suivantes :

- **Attendre ($e1$ et $e2$ ou $e3$)**: Elle permet d'attendre que : soit les événements $e1$ et $e2$ se produisent soit l'événement $e3$ se produise.
 - **Déclencher (ei)** : Elle permet de réveiller tous les processus en attente de cet événement si leurs conditions sont satisfaites et tous les événements attendus sont acquittés. Si aucun processus ne l'attend, l'événement ei est mémorisé.
- Donner une implémentation de ces deux primitives à l'aide des sémaphores

Correction de l'épreuve partielle

Exercice 1: (10 pts= 2*5)

- a- La structure Fork/Join est plus expressive de Parbegin Parend car cette dernière ne peut exprimer que des graphes de précédences proprement liés contrairement à la première qui peut exprimer tous les types de graphes de précédences.
- b- Le diagramme de temps d'exécution du programme:



Remarque : Les processeurs qui exécutent T4, T5 et T6 peuvent être différents de ceux qui exécutent les autres tâches.

- c- L'indivisible de l'instruction TAS sur un multiprocesseur est mise en œuvre en verrouillant le bus de données pour ne laisser qu'un seul processus à la fois accéder à la variable partagée, paramètre de cette instruction, lors de la tentative de son exécution simultanée par plusieurs processus.
- d- Pour surmonter la rigidité de leurs utilisations, les événements ont été étendus en considérant l'attente de la vérification d'une expression booléenne à base d'un ensemble d'événements.
- e- Dans le modèle Lecteurs/Rédacteurs, l'expression '*sans priorité explicite*' signifie que la condition d'accès d'un processus lecteur/rédacteur à la ressource partagée ne considère que la cohérence de celle-ci.

Exercice N°2 : (5 pts)

Implémentation

S : semaphore := 0 ; e :Tableau [1..3] de entier := 0 ; cpt : entier := 0 ; $mutex$: semaphore := 1

Primitive Attendre ($e1$ et $e2$ ou $e3$) ;

Debut

$P(mutex)$

Si ((($e1=0$) ou ($e2=0$)) et ($e3=0$)) **Alors**

$cpt := cpt + 1 ; V(mutex) ;$

$P(S)$

Sinon

$e1 := 0 ; e2 := 0 ; e3 := 0 ; V(mutex)$

Fsi

Fin ;

Primitive Declencher ($e[j]$) ;

Debut

$P(mutex) ;$

Si ($e[j] < 1$) **Alors**

$e[j] := 1 ;$

Si ($j=3$) ou ($j=1$) et ($e[2]=1$) ou ($j=2$) et ($e[1]=1$)

Alors

Si ($cpt < 0$) **Alors**

Tant que ($cpt < 0$) **Faire**

$V(S) ; cpt := cpt - 1 ;$

Fait ;

$e[1] := 0 ; e[2] := 0 ; e[3] := 0$

Fsi

Fsi

Fsi

Fin ;

La primitive *Declencher()* peut aussi s'écrire comme :

Primitive *Declencher* (*e[j]*) ;

Debut

P(mutex) ;

Si (*e[j]* <> 1) **alors**

e[j] := 1 ;

Si (((*e1* <> 1) **et** (*e2* <> 1)) **ou** (*e3* <> 1))

Alors

Si (*cpt* <> 0)

Alors Tant que (*cpt* <> 0) **Faire**

V(S) ; *cpt := cpt - 1* ;

Fait ;

e[1] := 0 ; *e[2] := 0* ; *e[3] := 0*

Fsi

Fsi

Fsi

Fin ;