**Université Aboubekr BELKAID Tlemcen**



**Faculté des Sciences**

**Département d’informatique**

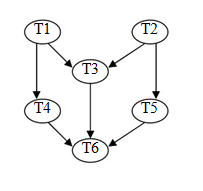
**Classe LMD S5 Informatique**

**Année Universitaire : 2021/ 2022**

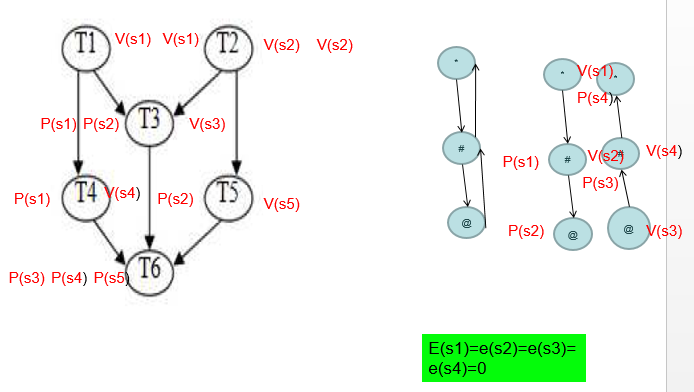
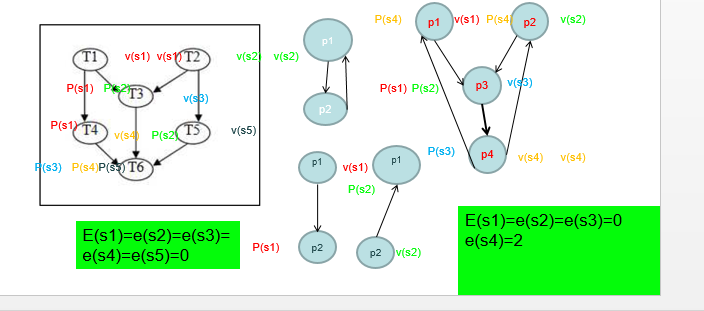
**Solution TD N°4 : LES SEMAPHORES**

**Exercice N°1**

Synchroniser les taches de graphe de précédence (graphe de causalité) suivant en utilisant les sémaphores :



**Solution exercice N°1 :**



**Graphe 2**

**E(s1)=0**

**E(s2)=0**

**E(s3)=2**



**P(s1) p(s2)**

**P(s3)**

**P(s3)**

**V(s1)**

**V(s2)**

**V(s3)v(s3)**

**Graphe 3**

**E(s1)=0**

**E(s2)=1**

**E(s3)=1**



**P(s1)**

**P(s1)**

**V(s1) v(s1)**

**P(s2) p(s3)**

**V(s2)**

**V(s3)**

**Exercice N° 2**

On a 3 processus p1, p2 et p3 concurrents. Le code du processus pi (i = 1,2,3) est le suivant :

Processus pi

{

printf (‘’ je suis le processus %d’’, i) ;   
}

On veut montrer sur l’écran la sortie suivante :

- je suis le processus 1

- je suis le processus 2

- je suis le processus 3

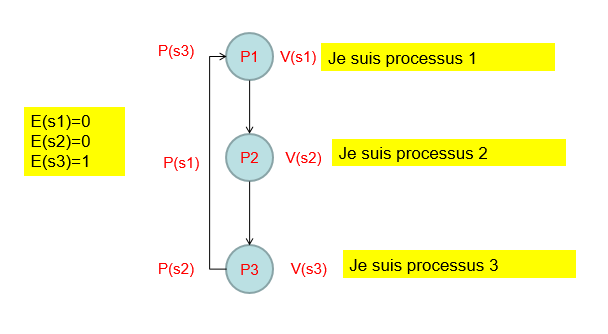
- je suis le processus 1

- je suis le processus 2

- je suis le processus 3

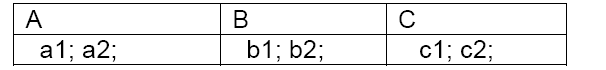
**Q1 :** synchroniser les 3 processus à l’aide des sémaphores

**Solution exercice N°2**



**Exercice N°3**

Considérez les 3 processus A, B et C concurrents suivants :



Où ai, bi et ci, pour i=1,2, sont des actions atomiques.

**Q1)** Synchronisez, en utilisant les sémaphores, les processus A, B et C s'exécutent en exclusion mutuelle

**Q2)** Utilisez les sémaphores pour que au plus 2 processus sur 3 s’exécutent en concurrence.

**Q3)** Utilisez les sémaphores pour forcer l’exécution des actions du processus A avant celles de B et C. Les actions de B et C s’exécutent en concurrence.

**Solution exercice N°3**

**Q1/ A , B et C s’exécutent en exclusion mutuelle**

**e(s)=1**

**P(s) A v(s)**

**p(s) B v(s)**

**p(s) C v(s)**

**Q2/ Deux processus au plus s’exécutent en parallèles**

**e(s)=2**

**p(s) A v(s)**

**p(s) B v(s)**

**p(s) c v(s)**

**Q3/**

**Solution 1 :**

**e(s)=0**

**A v(s) v(s)**

**p(s) B**

**p(s) C**

**Solution 2:**

**e(s1)=0**

**e(s2)=0**

**A v(s1) v(s2)**

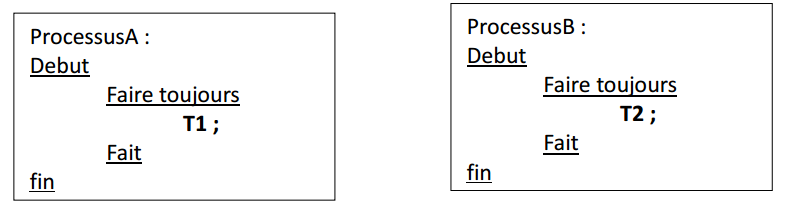
**p(s1) B**

**p(s2) C**

**Exercice N°4**

Soit l’exécution parallèle des deux processus suivants :

Debut  
ProcessusA ;



ProcessusB ;

Fin

Q1 : Utilisez un sémaphore pour synchroniser les 2 processus de telle manière que l’exécution de la tâche T1 ne soit jamais simultanée avec l’exécution de la tâche T2.

Q2 : Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T1T2T1T2...

Q3 : Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T2T1T2T2T1T2T2...

**Solution exercice N°4**

**Q1/**

**T1 s'exécute en exclusion mutuel avec T2**

**e(s)=1**

**p(s) T1 v(s)**

**p(s) T2 v(s)**

**Q2/**

**Ordre d’exécution : T1T2 T1T2 T1T2**

**e(s2)=1**

**e(s1)=0**

**p(s2) T1 v(s1)**

**p(s1) T2 v(s2)**

**Q3 / ordre d’exécution : T1 T2T2 T1 T2T2 T1 T2T2**

**e(s1)=0**

**e(s2)=2**

**p(s2) p(s2) T1 v(s1) v(s1)**

**p(s1) T2 v(s2)**

**D’autre exemple :**

**1/exécution T2T2 T1T1 T2T2 T1T1**

**e(s1)=4**

**e(s2)=0**

**p(s2)p(s2) T1 v(s1) v(s1)**

**p(s1)p(s1) T2 v(s2) v(s2)**

**-----------------------------------------------------**

**2/ t1 t2t2t2 t1 t2t2t2**

**e(s1)=0**

**e(s2)=3**

**p(s2)p(s2) p(s2) T1 v(s1)v(s1)v(s1)**

**p(s1) T2 v(s2)**

**----------------------------------------------------------**

**3/ t1t2 t3t3 t1 t2 t3t3**

**e(s1)=0**

**e(s2)=0**

**e(s3)=2**

**p(s3)p(s3) T1 v(s1)**

**p(s1) T2 v(s2)v(s2)**

**p(s2) T3 v(s3)**

**--------------------------------------------------**

**4/ t1t3t3 t2 t1 t3t3 t2**

**e(s1)=0**

**e(s2)=1**

**e(s3)=0**

**p(s2) T1 v(s1) v(s1)**

**p(s3) p(s3) T2 v(s2)**

**p(s1) T3 v(s3)**

**-------------------------------------------**

**5/ t1 t2 t2 t3 t3 t3 t4 t4 t1 t2 t2 t3 t3t3 t4 t4**

**e(s1)=0**

**e(s2)=0**

**e(s3)=0**

**e(s4)=2**

**p(s4)p(s4) T1 v(s1) v(s1)**

**p(s1) T2 v(s2) v(s2) v(s2)**

**p(s2) p(s2) T3 v(s3) v(s3)**

**p(s3) p(s3) p(s3) T4 v(s4)**

**-------------------------------------------------------**

**6/ t3t3t3 t1 t1 t2t2 t3t3t3 t1t1 t2t2**

**e(s1)=0**

**e(s2)=6**

**e(s3)=0**

**p(s3) p(s3) p(s3) T1 v(s1) v(s1)**

**p(s1) p(s1) T2 v(s2) v(s2) v(s2)**

**p(s2) p(s2) T3 v(s3) v(s3)**

**--------------------------------------------------------------**

**7/ t1t4t4 t2t2 t3 t1 t4t4 t2t2 t3**

**e(s1)=0**

**e(s2)=0**

**e(s3)=1**

**e(s4)=0**

**p(s3) T1 v(s1) v(s1)**

**p(s4) p(s4) T2 v(s2)**

**p(s2) p(s2) T3 v(s3)**

**p(s1) T4 v(s4) v(s4)**

**-----------------------------------------------------------**

**8/**

**T1T1T1 T2T2 T3T3T3 T1T1T1 T2T2 T3T3T3**

**e(s1)=0**

**e(s2)=0**

**e(s3)=9**

**p(s3) p(s3) p(s3) T1 v(s1) v(s1)**

**p(s1) p(s1) p(s1) T2 v(s2) v(s2) v(s2)**

**p(s2) p(s2) T3 v(s3) v(s3) v(s3)**

**-----------------------------------------------------------------------**

**9/ t1 t2 t3 t1t1 t2t2 t3t3 / t1 t2 t3 t1t1 t2t2 t3t3 /**

**#include <pthread.h>**

**#include <stdio.h>**

**#include <semaphore.h>**

**/\*\*La solution de l'affichage T1T2T3 T1T1T2T2T3T3 T1T2T3 T1T1T2T2T3T3\*\*/**

**sem\_t s1, s2 , s3 ,s4,s5,s6;**

**int a=1;**

**void \*T1(void \*arg)**

**{**

**for(int i=0;i<40;i++)**

**{**

**if(a)**

**{**

**sem\_wait(&s3);**

**printf("T1 ");**

**sem\_post(&s1);**

**}**

**else**

**{**

**sem\_wait(&s6);**

**sem\_wait(&s6);**

**printf("T1 ");**

**sem\_post(&s4);**

**sem\_post(&s4);**

**}**

**}**

**return (NULL);**

**}**

**void \*T2(void \*arg)**

**{**

**for(int i=0;i<40;i++)**

**{**

**if(a)**

**{**

**sem\_wait(&s1);**

**printf("T2 ");**

**sem\_post(&s2);**

**}**

**else**

**{**

**sem\_wait(&s4);**

**sem\_wait(&s4);**

**printf("T2 ");**

**sem\_post(&s5);**

**sem\_post(&s5);**

**}**

**}**

**return (NULL);**

**}**

**void \*T3(void \*arg)**

**{**

**for(int i=0;i<40;i++)**

**{**

**if(a)**

**{**

**sem\_wait(&s2);**

**a=0;**

**printf("T3 ");**

**sem\_post(&s3);**

**}**

**else**

**{**

**sem\_wait(&s5);**

**sem\_wait(&s5);**

**a=1;**

**printf("T3 ");**

**sem\_post(&s6);**

**sem\_post(&s6);**

**}**

**}**

**return (NULL);**

**}**

**int main ( )**

**{**

**sem\_init(&s1,0,0);**

**sem\_init(&s2,0,0);**

**sem\_init(&s3,0,1);**

**sem\_init(&s4,0,0);**

**sem\_init(&s5,0,0);**

**sem\_init(&s6,0,2);**

**pthread\_t threadA;**

**pthread\_t threadB;**

**pthread\_t threadC;**

**pthread\_create(&threadA,NULL,T1,NULL);**

**pthread\_create(&threadB,NULL,T2,NULL);**

**pthread\_create(&threadC,NULL,T3,NULL);**

**pthread\_join(threadA,NULL);**

**pthread\_join(threadB,NULL);**

**pthread\_join(threadC,NULL);**

**return 0;**

**}**

**Exercice N°4**

**Vous avez les 4 threads suivants : T1, T2, T3 et T4**

#include<stdio.h>

#include<stdlib.h>   
#include<pthread.h>   
#include<semaphore.h> sem\_t s1,s2,s3,s4;

**void \*T1(void \*arg)** {

**printf("L");**

**printf("M");   
printf("D");   
printf(" ");**

pthread\_exit(NULL); }

**void \*T2(void \*arg)** {

**printf("E");**

**printf("X");   
printf("A");**

pthread\_exit(NULL); }

**void \*T3(void \*arg)** {

**printf("S");**

**printf("5\n");**

pthread\_exit(NULL); }

**void \*T4(void \*arg)** {

**printf("M");**

**printf("E");   
printf("N");   
printf(" ");**

pthread\_exit(NULL); }

**Q1 :** synchronisez les 4 threads : T1, T2, T3 et T4 en utilisant les primitives de sémaphore sem wait(...), sem post(...) pour que le programme affiche obligatoirement **:   
EXAMEN LMD S5**

**T2 T4 T1 T3**

**Solution exercice N°5**

**Ordre d’execution est :**

**EXAMEN LMD S5 :**

**T2 T4 T1 T3**

**E(s1)= 0, e(s2)= 0, e(s4) =0,**

**P(s4) [ T 1: LMD] v(s1)**

**[ T2 : EXA] v(s2)**

**P(s1) [T3 : S5]**

**P(s2) [ T4 : MEN ] v(s4)**