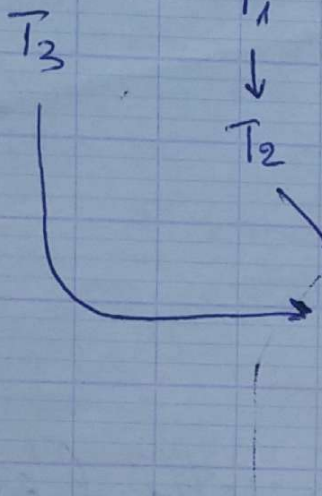
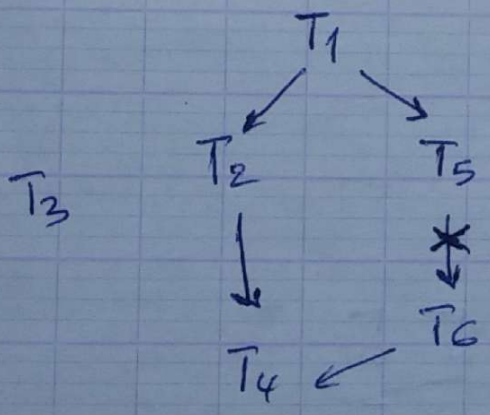
## Exo ① :




$(T_3 \parallel \{T_1, (T_2 \parallel \{T_5, T_6\})\}); T_4$

```
begin
  parbegin T3 ;
    begin T1
      parbegin T2
        begin T5 , T6 end
      parend
    end
  parend
  T4
end
```

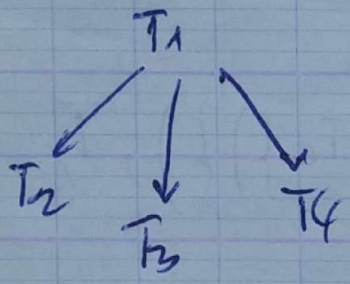
→ Ce graphe est proprement lié GPL




$(T_3 \parallel \{T_1, \{T_2 \parallel \{T_5, T_6\}\}, T_4\})$

```
begin
  parbegin T3
    begin
      T1
      parbegin T2
        | begin T5, T6 end
      parbeg
        T4
    end
  parend
end
```

①




```
Begin
   T1 .
   fork L3
   fork L4
L3: T2 , quit
L4: T3 , quit
   T4
end
```

② Join : il realise le rendez-vous. il empeche l'exec de l'inst qui suit join tant que n<>0.

n--; Si n ≠ 0 alors quit fsi.

→ le join doit être une inst indivisible.

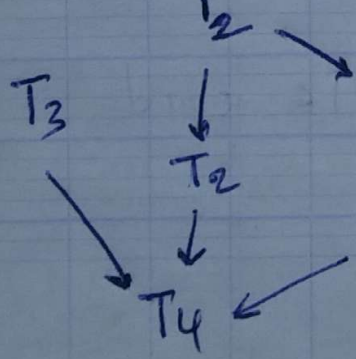
P1 → P2

T1 T2

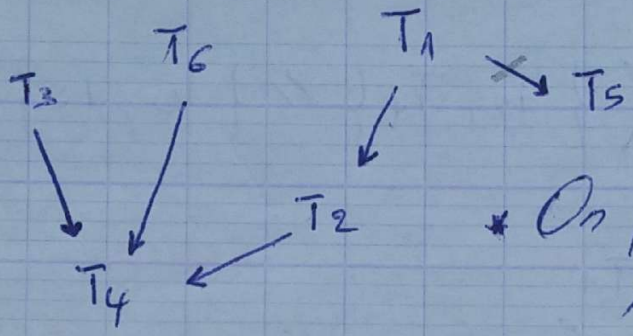
T3

```
begin  n=2
   fork L2
   T1 . goto L3

L2 . T2
L3 : join n
   T3
```




```
Begin  n=3 ;
fork L3
T1
fork L5
T2 , goto L4
L5 : T5, T6 , goto L4
L3 : T3
L4 : join n
   T4
End
```



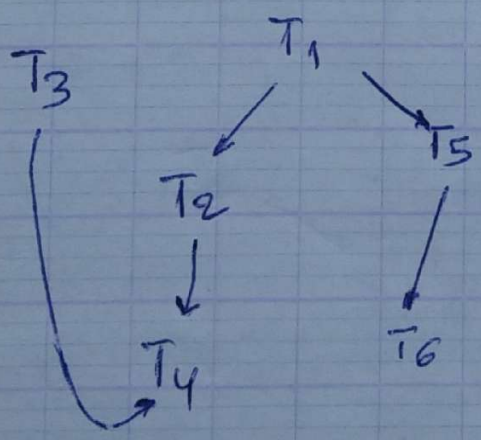

G n'est pas PP.

* On peut le rendre PL en Supp un arc. par exp $(T_1, T_5)$.

$$\left( T_5 \,//\, \left\{ \left( T_3 \,//\, T_2 \,//\, (T_1, T_2) \right) ; T_4 \right\} \right)$$

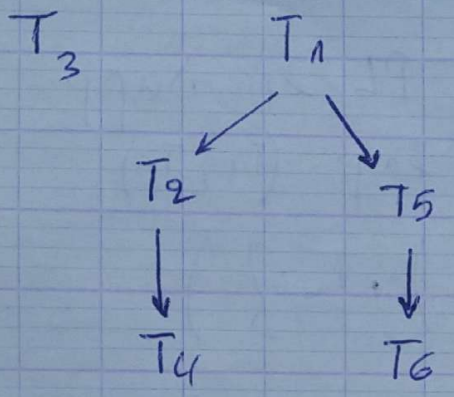
```
begin
  parbegin T5
    begin
      parbegin T3, T2
        begin T1, T2 end
      parend
      P4
    end
  parend
end
```

$G_4$ $(T_6, T_4)$




$T_4$ en chevauchant entre deux blocs tels que (l'un des noeuds des deux blocs > $T_4$).

G5 $(T_3 = \{T_1 ; (\{T_2, T_4\} // \{T_5, T_6\})\})$




```
par begin T3
    begin T1
        par begin
            begin T2, T4 end
            begin T5, T6 end
        par end
    end
par end
end
```

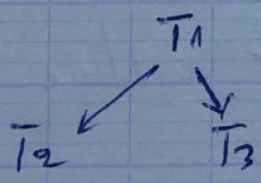
---

② L'outil fork/join : Cet outil exprime tout type de graphe "PL ou non PL"

2.1 fork :

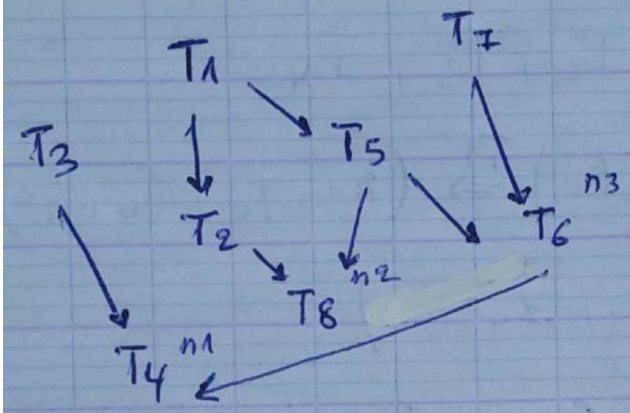
fork $T_i$ et et 1 $T_j$

→ fork permet de Créer 2 taches




```
Begin
    T1,
    fork L3
        T2 qui-
    end
```

pour n taches
On a besoin
de n-1 fork

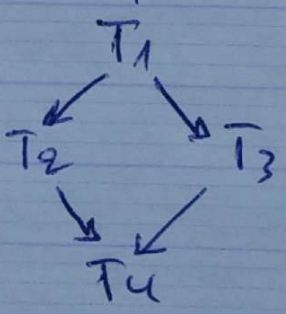



```
Begin  n1 = 2 ; n2 = 2 ; n3 = 2
       fork L7
       fork L3
       T1;
       fork L5
       T2, goto L8
L5     T5
       fork L6
L8     join n2
       T8, quit
L7     T7
L6     join n3
       T6, goto L4
L3     T3
L4     join n1, T4
```

⑤ Les Comportements possibles d'1 Syst de tâches:

Dans un Syst de tâches ou ∃ le parallélismes on peut avoir plsrs Comportemnt.

pour autoriser pls Comportmnt il faut assurer que le resultat soit unique.




T1d T1m T1F T2d T3d T2m

$T_1$: line(x);
$T_2$: $y = x * 2$;
$T_3$: $z = y + 10$;
$T_4$: $T = y * z$;

$C_1 = T_1, T_2, T_3, T_4$

$(x_0, y_0, z_0, t_0) \xrightarrow{T_1} (x_1, y_0, z_0, t_0)$

$\xrightarrow{T_2} (x_1, 2x_1, z_0, t_0)$

$\Rightarrow T_3 (x_1, 2x_1, 2x_1 + 10, t_0)$

$\xrightarrow{T_4} (x_1, 2x_1, 2x_1 + 10, 4x_1^2 + 20x_1)$

$C_2 = T_1, T_3, T_2, T_4$

$(x_0, y_0, z_0, t_0) \xrightarrow{T_1} (x_1, y_0, z_0, t_0) \xrightarrow{T_3} (x_1, y_0, y_0 + 10, t_0)$

$\xrightarrow{T_2} (x_1, 2x_1, y_0 + 10, t_0) \xrightarrow{T_4} (x_1, 2x_1, y_0 + 10, 2x_1 y_0 + 20x_1)$ -

# Serie ②:

## Exo ②:

→ Cond d'entrée pour $P_i$ : $D_j$ = faux.

→ invariant. $D_i$ = Vrai.

① EM:

a) $P_j$ en SC $P_i$ desire d'entrée risque t'il de le joindre.

$P_j$ en SC ⟹ $D_j$ = Vrai ceci empeche $P_i$ d'entrée en SC "Cond non verifié"

b) SC libre et $P_i$, $P_j$ desire d'entrer risquent t'il d'entrer en même tmp ⟹ $D_i$ = $D_j$ = Vrai.

* 1er Cas tour = j : $P_i$ retire sa demande $D_i$ = F
- attend la boucle "attente jusq $P_j$ entre et sort de la SC"

② BM:

- $P_i$, $P_j$ desirent d'entrée risquent d'attendre indf. d'après (1b) BM est evité.

4) AB : $P_j$ en SC $P_i$ désire rentrer.

risque t'il d'attendre indéf si $P_j$ continue à sortir et dmd la SC.

⇒ $P_j$ en SC alors $D_j$ = Vrai ;

$P_i$ en attente → B1 ⇒ $D_i$ = Vrai, tour = i
→ B2 ⇒ $D_j$ = faux, tour = j,

a) $P_i$ en attent ds B1 ⇒ $D_i$ = Vrai, tour = i.
$P_j$ sort ⇒ $D_j$ = faux et tour = i.
$P_j$ redemande ⇒ $D_j$ = Vrai.
→ $P_j$ retire sa dmd $D_j$ = faux. $P_i$ rentre.

b) $P_i$ en attente ds B2 ⇒ $D_j$ = f et tour = i
$P_j$ sort ⇒ $D_j$ = faux et tour = i.
$P_j$ redemande ⇒ $D_j$ = Vrai
* Mais lorsque $P_i$ observe tour = i sort de B2 et att ds B1

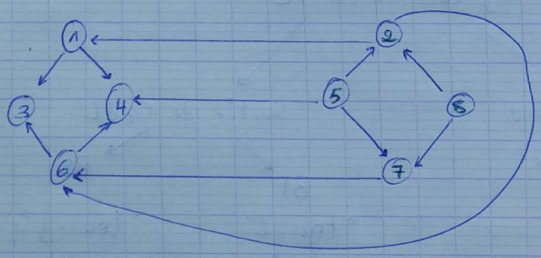
⇒ AB assuré

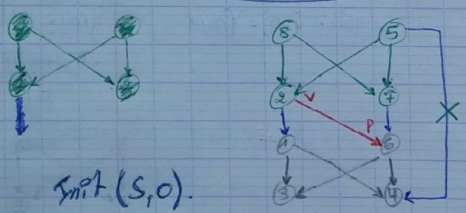
③ Progression : $P_j$ loin de SC. $P_i$ désire rentrer
risque t'il d'attendre indéf ?

$P_j$ loin de la SC
→ n'est pas venu
" $D_j$ = faux "
→ sortie de la SC.
→ B1
" tour = i "
* grâce à $D_j$ = faux.
→ B2
" tour = j "
* grâce à tour = i et rentre ds B1

Serie ③:

Exo ①:





Init (S,0).

Begin

parbegin $T_6$, $T_5$ parend

parbegin

* begin $T_2$, V(S), $T_1$ end

begin $T_7$, P(S), $T_0$ end

parend

end parbegin $T_3$, $T_4$ parend.

* Si j'avais parbegin $T_2$, V(S) parend → faux car il faut une Seq entre $T_2$ et V(S).

Exo ②:

| P1 | P2 | P3 |
|---|---|---|
| while (1) { | while (1) { | while (1) { |
| P(S1); | P(S2); | P(S3); |
| X = X + Z + 2; | Y = X + 10; | Z = Y * 2 + 1; |
| V(S2); } | V(S3); } | V(S1); } |

① 

| | S1 | S2 | S3 | X | Y | Z |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 |
| ✓ | 0 | 0 | 1 | 0 | 0 | 1 |
| ✓ | 0 | 1 | 0 | 0 | 10 | 0 |
| | 0 | 1 | 1 | 0 | 10 | 21 |
| ✓ | 1 | 0 | 0 | 2 | 0 | 0 |
| | 1 | 0 | 1 | 23 | 0 | 1 |
| | 1 | 1 | 0 | 2 | 10/12 | 0 |
| | 1 | 1 | 1 | Plusts | | |

→ Solution Soit : { S1 = 0 ; S2 = 0 ; S3 = 1. / S1 = 1 ; S2 = 0 ; S3 = 0. / S1 = 0 ; S2 = 1 ; S3 = 0.

② La solution est incorrecte parceque on peut avoir plusieurs résultat.

* Table de Bernstein.

| | R | W | |
|---|---|---|---|
| $T_1$ | z, x | x | $T_1$ ✕ $T_2$ |
| $T_2$ | x | y | $T_2$ ✕ $T_3$ |
| $T_3$ | y | z | $T_1$ ✕ $T_3$ |

Exo ③ :

* Structure de données :
  CapR : entier init à 10
  SC, SV : Sémaphores privés init (SC, 0). init (SV, 0).
  mutex : Sémaphore binaire protégé l'accès à CapR.
  attc = 0, attv = 0 : entier ; et attc :

P Cam
Debut
  P(mutex) ; // 1 Cam qui peut acced à CapR.
  Si (CapR ⩾ 8) alors CapR -= 8 ; Cam = Vrai ;
                     V(mutex) ;
  Sinon attc++ ; V(mutex) ; P(SC) ;
  fsi
  < Accès au pont >
  P(mutex) ; Cam = faux ; CapR += 8 ;
  Si (attc ≠ 0) alors V(SC) ;
                      Cap_R -= 8 ;
                      attc -- ;
                      Cam = Vrai // on a un Cam sur pont
  Sinon
    tq (attv ≠ 0) et (CapR ⩾ 2) et (attc = 0)
    faire V(SV) ; attv -- ; CapR -= 2 ;
          V(mutex) ; P(mutex) ; // pour q1 nouv Cam ait la main ds PSm
    fait
    V(mutex) ;
  fi

// Processus des Voitures :
Debut
  P(mutex) ;
  Si (Cap_R ⩾ 2 et attc = 0) ou (CapR = 2 et Cam = V)
    alors CapR -= 2 ; V(mutex) ;
  Sinon attv++ ; V(mutex) ; P(SV) ;
  fi
  < Accès au Pont >
  P(mutex) ; CapR += 2 ;
  si (attc ≠ 0) alors
    si (capR ⩾ 8) alors
      CapR -= 8 ; attc -- ;
      Cam = V ; V(SC) ;
    fi fq ?
  Sinon si (attv ≠ 0) alors CapR -= 2 ; attv -- ;
                            V(SV) ;
  fi
  V(mutex) ;
Fin

Exo 4:

SD:

S$, SM, SL : Sem privés

N$, NM, NL : entiers à zéro

atti, attm, attl : entiers à zéro.

mutex : Sém binaire.

Proc Eng

Debut

P(mutex);

Si (NM ≠ 0) ou (NL ≠ 0) alors

attl++; V(mutex); P(S$);

Sinon

N$++; V(mutex);

fsi

<accès au biblio>

P(mutex); N$--;

Si (n$ -- 0) alors

Si (attl ≠ 0) alors

tq (attl ≠ 0) alors

tq (attl ≠ 0) et (atti = 0)

faire attl--; V(SL); NL++;

V(mutex); P(mutex);

fait

Sinon

tq (attm ≠ 0) et (atti == 0) et (attl == 0)

faire attm--; NM++; V(SM);

V(mutex); P(mutex);

fait

fsi

V(mutex);

fsi

Fin

→ Proc Licence

Debut P(mutex);

Si (N$ ≠ 0) ou (NM ≠ 0) ou (atti ≠ 0) alors

## ② file d'attente:

SD: S: sem privé init à 0.

F: file d'entier { 0 → master, 1 → Licence, 2 → ING }

N[0..2]: Tab d'entier N[0] → NM, N[1] → NL, N[2] → NI.

mutex : Sém binaire "pour protéger file et Tab"

```
Pmaster
Debut
    P(mutex);                    (i+1)%3        (i+2)%3
    Si (Vide(F)) ou (N[1]≠0) ou (N[2]≠0) alors
        enfiler(F,0); V(mutex); P(S);
    Sinon
        N[0]++; V(mutex);
    fsi
    <accès au biblio>            P(mutex);
                                 N[0]--;
    Si (N[0]==0) alors
        Si (Vide(F)) alors       j = tete[F];
            tq (Vide(F)) et (tete(F)==j)
            faire defiler(F);
                  V(S); N[j]++;
            fait
        fsi
    fsi
    V(mutex);
Fin
```