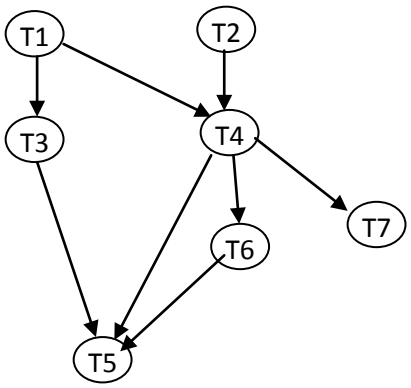
Faculté des sciences exactes
Département d'Informatique
Module : **Système d'exploitation 2**
Enseignant : **M. ATMANI**

Exercices supplémentaires

Exercice1 (Fork/Join)

Donner le programme parallèle en utilisant les primitives fork/join du graphe de précédence ci-contre.



Solution

1) Le nombre de jointures qu'on peut avoir est 2 (la première va joindre 2 exécutions parallèles et la deuxième va joindre 3 exécutions parallèles). Donc, N1 = 2 et N2=3
2) Au début, on remarque qu'on a deux tâches parallèles (T1 et T2), ce qui nécessite la création d'un processus fils1 avec la primitive ***fork*** pour qu'il exécute l'une des deux tâches (T1 et T2). Après la création du fils1, le père peut exécuter sa tâche.
3) Le reste du programme est donné comme suit :

```
N1 = 2 ; N2=3 ;
        Fork E1 ;
        T1 ;
        Fork E2 ;
        T3 ;
        Goto E5 ;
E1 : T2 ;
E2 : join N1 ;
        T4 ;
        Fork E3 ;
        Goto E5 ;
E3 : fork E4 ;
        T6 ;
        Goro E5 ;
E4 : T7 ;
        Quit ;
E5 : join N3 ;
        T5 ;
```

Exercice2 (Moniteurs)

Le problème de partage d'une base de données ou d'un fichier entre plusieurs processus est connu sous l'appellation du ***problème des lecteurs / rédacteurs*** (P.J. COURTOIS, 1971). En fait, **les lecteurs** consultent le fichier sans en modifier le contenu ; les rédacteurs quant à eux peuvent en modifier le contenu.

Dans cet exercice on veut synchroniser l'accès à un fichier partagé entre les lecteurs et les rédacteurs, en respectant certaines conditions :

- Plusieurs lecteurs peuvent lire le fichier en même temps ;
- Un seul rédacteur à la fois peut écrire dans le fichier ;
- A la sortie d'un rédacteur, s'il existe des lecteurs et des rédacteurs bloqués, la priorité est donnée aux lecteurs ;

Question : écrivez le programme avec les moniteurs.

Soit ***lec_red*** un moniteur chargé d'assurer le respect des contraintes de priorité aux lecteurs. On impose la forme suivante aux accès au fichier.

Processus Lecteur	**Processus Rédacteur**
lec_red.*debut_lire()* ;	***lec_red***.*debut_ecrire();*
<Accès en lecture>	<accès en écriture>
lec_red.*fin_lire() ;*	***lec_red***.*fin_ecrire() ;*

Les procédures *debut_lire*, *fin_lire*, *debut_ecrire*, *fin_ecrire* doivent assurer le respect des contraintes de franchissement exprimées ci-dessus. Dans notre exemple, la priorité est donnée aux lecteurs (une écriture n'est pas autorisée si des lecteurs sont en attente). On définit les variables suivantes :

ecr = une écriture est en cours (booléen)

nl = nombre de lecteurs en attente ou en cours de lecture.

Les conditions de franchissement s'expriment alors ainsi :

Franchissement (lecture) : ┐ ecr (pas d'écriture en cours)

Franchissement (écriture) : ┐ ecr et nl=0 (ni écriture ni lecture en cours, pas de lecture en attente)

```
Moniteur lec_red;

    Var     ecr: booleen;
            nl : entier;
            c_ecr, c_lect: condition;
Début
    Procedure debut_lire()
    Debut
        nl :=nl+1 ;
        si ecr alors
            c_lect.wait ;
        fsi
        c_lect.signal;   //réveil en chaîne
                         // des lecteurs
    Fin;
    Procedure fin_lire()
    Debut
        nl=nl-1 ;
        si nl= 0 alors
            c_ecr.signal ;   // le dernier
                             //lecteur a fini
        fsi
    Fin
```

```
    procedure debut_ecrire()
     Début
        si ecr ou nl >0 alors
            c_ecr.wait ;
            // Écriture ou lecture en cours
        fsi
        ecr := vrai ;
    Fin
    procedure fin_ecrire()
    Début
        ecr :=faux ;
        si nl >0 alors
            c_lect.signal ;
       Sinon
            c_ecr.signal ;
        fsi
    Fin

    Début       // initialisation
        ecr := faux ;
        nl := 0 ;
    Fin
Fin file.
```