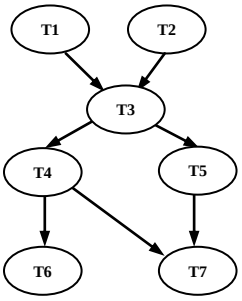
Système d'exploitation 2

Exercice N° : 01 *(04 pts)*

1. Donnez un exemple d'un système de tâche indéterminé
2. Certains signaux ne peuvent être ni capturés, ni ignorés. Citez deux ?

Exercice N° : 02 *(04 pts)*

Transformer le graphe de précédence ci-dessous en un algorithme en utilisant ParBegin, ParEnd et les sémaphores si nécessaire:



Exercice N° : 03 *(04 pts)*

Ecrire un programme en C dont le thread principal (main) crée un thread qui lit un nombre entier, donné par l'utilisateur, et le retourne au thread principal qui l'affiche.

Exercice N° : 04 *(04 pts)*

Supposons qu'il y a trois processus concurrents Processus_A, Processus_B et Processus_C qui affichent continuellement "A", "B" et "C" respectivement.

```
Processus_A
Repeat
 Afficher('A') ;
Until false
```

```
Processus_B
Repeat
Afficher('B') ;
Until false
```

```
Processus_C
Repeat
Afficher('C') ;
Until false
```

En utilisant **seulement les sémaphores** pour coordonner l'affichage, donner pour chacun des cas suivant le pseudo-code des trois processus:

1. Le nombre de "C" affichés soit toujours inférieur ou égal à la somme des nombres de "A" et de "B" affichés.
2. La chaine de caractères affichée soit : (C.B.A)*. Exemple : CBACBACBACBA...etc

Exercice N° : 05 *(04 pts)*

Ecrire un programme qui affiche les nombres compris entre 0 et 20 au moyen de deux processus (le processus père affiche les nombres pairs, le fils affiche les nombres impairs). Les deux processus se synchronisent par les signaux afin d'assurer l'affichage des nombres dans un ordre correcte (0, 1 , 2,…..20) .

BON COURAGE

Système d'exploitation II (Corrigé type)

Exercice N° : 01 *(04 pts)*

1. Un système qui contient au moins 2 tâches interférentes.
2. SIGKILL, SIGSTOP

Exercice N° : 02 *(04 pts)*

Debut
Sémaphore S init à 0
ParBegin T1, T2 *ParEnd*
T3
ParBegin
Begin T4, signal(S), T6 End
Begin T5, wait(S), T7 End
ParEnd
Fin

Exercice N° : 03*(04 pts)*

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>


void *Mythread(){
    int val;
    printf("entrer un nombre:\n");
    scanf("%d",&val);
    pthread_exit((void *)val);
}


int main()
{
    int valeur;
    pthread_t tid;
    pthread_create(&tid, NULL, Mythread, NULL);
    pthread_join(tid, (void *)&valeur);
    printf("la valeur est: %d",valeur);
    return 0;
}
```

Exercice N° : 04 ***(04pts)***

1/ Sémaphore : S initialisé à 0.

Processus_A	Processus_B	Processus_C
Repeat *Afficher('A') ;* **Signal(S);** **Until false**	**Repeat** *Afficher('B') ;* **Signal(S);** **Until false**	**Repeat** **Wait(S);** *Afficher('C') ;* **Until false**

2/ Sémaphores : S3 initialisé à 1, S1, S2 initialisés à 0.

Processus_A	Processus_B	Processus_C
Repeat ***Wait(S1);*** *Afficher('A') ;* **Signal*(S3);*** **Until false**	**Repeat** ***Wait(S2);*** *Afficher('B') ;* **Signal*(S1);*** **Until false**	**Repeat** ***Wait(S3);*** *Afficher('C') ;* **Signal *(S2);*** **Until false**

Exercice N° : 05*(04 pts)*

```
#include <signal.h>
int c = 0;
void inc(int sig)
{
printf("j'ai reçu le signal SIGUSR1\n");
c += 2;
return;
}
void main(void)
{
int pid, ppid;
signal(SIGUSR1,inc);
if ((pid = fork()) == 0)
{
c = 1;
signal(SIGUSR1,inc);
ppid = getppid();
for(; c <= 19;)
{
sleep(5);
printf("%d\n",c);
kill(ppid,SIGUSR1);
pause();
}
else
for(; c <= 20;)
{
sleep(5);
printf("%d\n",c);
kill(pid,SIGUSR1);
pause();
}
return 0;
}
```