CORRIGE TEST TP : PARTIE THEORIQUE

Exercice1 (/5) : Répondre aux questions suivantes :
1) Donner la sortie du programme suivant?

#include <unistd.h>
#include <stdio.h>
 	
int main(void){
if (fork()) printf("a");
else{
if (!fork() printf("b");
else printf("c");
}
printf("d");
return 0;
}
	adcdbd (**1**)
	justification : (**1**)
	Le processus père précède fils :
	le 1er fork : père affiche a puis d
	else : (le 1er fils) : le deuxieme fork : père affiche c puis d
	et enfin le 2eme fils affiche b puis d

2) Peut-on utiliser la primitive `pthread_join()` dans un thread secondaire ?Justifier !

Le pthread_join est utilisé bloquer le thread en cours jusqu'à exécution finale du thread précisé dans le join !

Pthread_join() est utilisé dans thread principal pour attendre ses threads secondaires,

ou dans thread secondaire pour attendre un thread tertiaire qu'il a lui-même crée,

ou si **la déclaration des threads est global**, dans ce cas n'importe quel thread peut l'utiliser pour attendre un autre thread autre que lui-même. (**1**)

3) Pourquoi la primitive pthread_cond_wait ne peut s'exécuter qu'entre une pthread_mutex_lock et pthread_mutex_unlock ? Cela peut-il provoquer un interblocage ?

le variables conditions ont été crées pour les moniteurs donc doivent s'exécuter en EM, le lock (verrou) permet d'assurer cette EM. (**1**)

le pthread_cond_wait ne provoque pas d'interblocage puisque le lock est libéré avant que le processus ne soit enfilé ! et à son réveil (pthrad_cond_signal) il le verrouille. (**1**)

Il y'a interblocage si le signal est exécuté (aussi en EM) avant le wait, ou si on oublie le unlock !

Exercice2 (/5) : Soit le programme suivant:

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

int x =0;

void *fonc1(void *a){
int loca = * ( int *) a ;
loca=loca- x;
```

```
x=loca ;
printf("a = %d\n", loca);
return(NULL);
}

void *fonc2(void *b){
int locb =*( int *) b;    (0,25)
locb=locb + x;
x=locb ;
printf("b = %d\n", locb );
return(NULL);
}

int main ( ){
int a = 2 , b = 1;
pthread_t th1;
pthread_t th2;        (0,25)
pthread_create(&th1, 0, fonc1, &a);
pthread_create(&th2, 0, fonc2, &b);
pthread_join(th1);
pthread_join(th2);    (0,25)
printf("ici main, x = %d \n", x);
return 0;
}
```

2) Donner la sortie du programme (en précisant les incertitudes) ?

a=2 (pas sûr pour la valeur et la position) (**0,5**)

b=3 (pas sûr pour la valeur et la position) (**0,5**)

x=3 (pas sûr pour la valeur et sûr pour la position) (**0,5**)

3) Quel est le problème du programme et proposer une solution ?

Le programme n'est pas déterministe à cause de la variable x ! il nécessite une synchronisation, un verrou les fonctions verrouiller() et déverrouiller() vont protéger la variable x dans les deux threads secondaires ! (**0,5**)

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

int x =0;
pthread_mutex_t verrou=PTHREAD_MUTEX_INITIALIZER ; (0,25)

void *fonc1(void *a){
int loca = * ( int *) a ;
pthread_mutex_lock(&verrou);          (0,5)
loca=loca- x;
x=loca ;
pthread_mutex_unlock(&verrou);        (0,5)
printf("a = %d\n", loca);
return(NULL);
}
```

```
void *fonc2(void *b){
int loci =*( int *) b;
pthread_mutex_lock(&verrou);          (0,5)
locb=locb + x;
x=locb ;
pthread_mutex_unlock(&verrou);        (0,5)
printf("b = %d\n", locb );
return(NULL);
}

int main ( ){
int a = 2 , b = 1;
pthread_t th1;
pthread_t th2;
pthread_create(&th1, 0, fonc1, &a);
pthread_create(&th2, 0, fonc2, &b);
pthread_join(th1,NULL);
pthread_join(th2,NULL);
printf("ici main, x = %d \n", x);
return 0;
}
```

-FIN-