Série 1:Manipulation de threads et synchronisation

Exercice 1

1- Reprendre le programme suivant avec un éditeur de texte:

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

long compteur[3];

/* fonction executee par chaque thread */

void* fonc_thread(void *k)
{long j;
j=(long ) k;
printf("Thread numero %ld :mon tid est %ld\n",j,pthread_self());
for(;;) compteur[j]++;
}

main()
{
long i, num;
pthread_t pth_id[3];

/* creation des threads */
for(num=0;num<3;num++)
{
pthread_create(&pth_id[num], 0,fonc_thread,(void *) num);
printf("Main: thread numero %ldcreee: id = %ld\n",num,pth_id[num]);
}
```

```
sleep(1); /* attente de 1 s */
printf("Affichage des compteurs (20 fois)\n");
for(i=0;i<20; i++)
{
printf("%ld \t%ld \t%ld\n",compteur[0], compteur[1],compteur[2]);
sleep(1);
/* attente de 1 s entre 2 affichages */
}
exit(0);
}
```

2- Compiler le programme via la commande suivante:

gcc thread_exo1.c -o exo1 -lpthread

3- Que fait le programme?

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

long compteur[3];

/* fonction exécutée par chaque thread */

void* fonc_thread(void *k)
{int var;
long  j;
j=(long ) k;
printf("Thread numero %ld :mon tidest %ld\n",j,pthread_self());
for(var=0;var<20;var++) {compteur[j]++;printf("%ld \n",compteur[j]);}
}

main()
```

```
{
long i, num;
pthread_t pth_id[3];

/* creation des threads */
for(num=0;num<3;num++)
{
pthread_create(&pth_id[num], 0,fonc_thread,(void *) num);
printf("Main: thread numero %ldcreee: id = %ld\n",num,pth_id[num]);
}
exit(0);
}
```

4- Que se passe-t-il ? comment corriger ça ?

Exercice 2

Ecrire un programme qui permet à un thread de lire des caractères au clavier et les passer à un autre thread qui se charge de les afficher. Le thread principal (le père) se charge de la création de ses fils et de l'attente de leur mort. La fin des threads fils survient à l'arrivée du caractère "F".

Exercice 3

L'objectif de l'exercice est de réaliser un programme qui joue au jeu de pierre-feuille-ciseau. La règle du jeu est la suivante: deux joueurs annoncent en même temps l'un de ces trois objets. Le joueur gagne selon son annonce et celle de son adversaire: la pierre perd face à la feuille, mais gagne face aux ciseaux et les ciseaux gagnent face à la feuille. Si les deux joueurs annoncent le même objet, personne ne gagne. Le jeu se joue en NbTours.

1- Ecrire un programme qui lance 3 threads, un pour chaque joueur et le troisième pour compter les points en respectant les contraintes de synchronisation.

Exercice 4

On veut modéliser le problème du producteur/consommateur.

1- Ecrire les fonctions producteur et consommateur en respectant les contraintes de synchronisation. Le producteur a pour rôle de créer des ressources (génère des nombres

aléatoires) avec la fonction produire()et les place dans un tableau (variable partagée) et le consommateur prend les ressources et les consomme avec la fonction consommer().

2- Ecrire le programme principal qui lance plusieurs threads consommateurs et plusieurs threads producteurs et vérifier que le programme s'exécute correctement.

Exercice 5

Ecrire un programme qui lance trois threads.

Le premier thread demande une valeur à l'utilisateur et lui donne le choix d'afficher ses multiples (jusqu'à 50) ou bien ses diviseurs. Selon le choix effectué, la main est donnée à l'un des deux autres threads :

Le deuxième thread affichera les multiples du nombre

Le troisième thread affichera les diviseurs du nombre

1- On désire simuler à l'aide des sémaphores l'utilisation d'une unique piste d'aéroport. A caque instant il ne peut y avoir qu'un avion qui atterrit ou qui décolle. Votre programme acceptera deux paramètres: Le nombre d'avions devant décoller et Je nombre d'avions souhaitant atterrir. Votre processus devra créer simultanément autant de threads fils que d'avions à gérer. Pour effectuer la simulation chaque thread avion commencera par attendre un temps aléatoire (entre 1 et 10 secondes) et affichera un message signalant qu'il souhaite atterrir ou décoller ainsi que son numéro (ieme avion). On considèrera que le temps de décollage est de 5 secondes et celui d'atterrissage de 3 secondes. Un message devra signaler qui libère la piste à un instant donné .

2- Modifier le programme pour simuler deux pistes d'atterrissage avec la priorité à la première piste.