

Programmation Concurrente

Concurrence et non-déterminisme

Série 4

Exercice 1

Soit le pseudo code du programme P multi-threadé suivant :

```
int x = 2;

thread1 :
    while (1) {
        x = 1;
    }

thread2 :
    while (1) {
        x = 2;
    }

thread3 :
    while (1) {
        if (x == 2) exit(0);
    }
```

Est-il possible que P ne se termine jamais ?

Justifiez votre réponse.

Exercice 2

Soit le pseudo code du programme P multi-threadé suivant :

```
int y = z = 0;

thread1 :
    x = y + z;

thread2 :
    y = 1;
    z = 2;
```

Une fois P terminé, est-il possible que la valeur de la variable x soit égale à 2 ?

Développez votre réponse.

Exercice 3

Soit la boucle suivante:

```
for (int i = -10; i < 10; i++) {  
    x++;  
}
```

Sur une machine monoprocesseur, quelles sont les valeurs extrêmes (min, max) du domaine de valeurs possibles qu'est susceptible de prendre la **variable partagée x** , initialisée à 0, après exécution complète du code ci-dessus par deux threads distincts?

Développez vos réponses.

Exercice 4

Soit le code suivant :

```
#define MAX_STRING_SIZE 256  
  
char *strtoupper(char *string) {  
    static char buffer[MAX_STRING_SIZE];  
    int index;  
    for (index = 0; string[index]; index++) {  
        // on part du principe que toupper est MT-safe et  
réentrant  
        buffer[index] = toupper(string[index]);  
    }  
    buffer[index] = 0;  
    return buffer;  
}
```

- Le code est-il réentrant ? Justifiez votre réponse.
- Le code est-il thread-safe ? Justifiez votre réponse.
- Dans la négative, modifiez le code pour qu'il devienne réentrant **et** thread-safe. A noter que modifier l'interface est autorisé.