Université de Limoges | FACULTÉ DES SCIENCES ET TECHNIQUES

Master 1ère année

UE Parallélisme II

Examen — mai 2011

Durée : 1h45 — Documents autorisés

1 – 4pts Décrire le comportement du morceau de programme suivant (création de thread, détails du travail réalisé par chaque thread, synchronisation, *etc.*) :

```
#define TAILLE 100
...
{
        int n = TAILLE;
        int a[TAILLE];
        int b[TAILLE];
        #pragma omp parallel shared(n,a,b) private(i)
        {
                #pragma omp for schedule(static,25)
                for (i=0; i<n; i++)
                        a[i] = i;
                #pragma omp for schedule(static,25)
                for (i=0; i<n; i++)
                        b[i] = 2 * a[i];
        }
        ...
```

a. dans le cas où l'ordinateur dispose de 2 cœurs ;

b. dans le cas où l'ordinateur dispose de 4 cœurs.

c. Est-ce possible d'obtenir un comportement identique dans les 2 cas ? Et si oui, comment ?

2 – 4pts Traduire le morceau de programme C suivant pour qu'il utilise des threads POSIX au lieu de threads OpenMP :

```
#define TAILLE 50
...
{
        int n = TAILLE;
        int a[TAILLE];
        int b[TAILLE];
        #pragma omp parallel for shared(n,a,b) private(i) schedule(static,25)
        {
                for (i=0; i<n; i++)
                        b[i] = 2 * a[i];
        }
```

3 – 4pts Écrire un programme C utilisant OpenMP et réalisant, en parallèle la moyenne, des valeurs contenues dans un tableau de 50 flottants :

```
...
float mes_valeurs[50];
float ma_moyenne = 0.0;
```

4 – Soient les 4 threads suivantes, s'exécutant chacune une seule fois :
4pts

```
Thread A {          Thread B {          Thread C {          Thread D {
   P(s2);              P(s2);              P(s1);              P(s1);
   x = x + y;          x = x * y;          y = y * 5;          P(s3);
   V(s2);  }           V(s2);  }           V(s3);  }           x = x + 1;
                                                               V(s2);  }
```

Elles utilisent pour leur synchronisation trois sémaphores `s1`, `s2` et `s3`.
La valeur initiale de `s1` est 2, celle de `s2` est `0` et celle de `s3` est `0`.

a. Quelles sont les différentes valeurs possibles des variables `x` et `y` à la fin de l'exécution complète des quatres *threads*, si `x` est initialisé à `0` et `y` à `1` ?

b. Qu'est-ce que cela changerait si la valeur initiale de `s1` était 1, celle de `s2` à `0` et celle de `s3` à `1` ?

5 – Un programme a été décomposé en 5 tâches $\{t_1, t_2, t_3, t_4, t_5\}$.
4pts L'analyse du parallélisme pouvant être exploité entre ces différentes tâches est la suivante :

- ⋆ les tâches $\{t_1, t_3, t_4, t_5\}$ peuvent être réalisées en parallèle ;
- ⋆ la tâche t_2 ne peut être faite qu'après les tâches $\{t_1, t_3\}$;
- ⋆ les tâches t_4 et t_5 ne peuvent être exécutées qu'après la tâche t_2.

a. Donnez un graphe des dépendances temporelles pour ce programme.

b. Indiquez comment à l'aide de Sémaphore, en reprenant la notation de l'exercice 4, vous pouvez synchroniser les tâches entre elles :

```
Thread i {
  P(s);
  // travail ti
  V(s);  }
```

c. Comment à l'aide d'OpenMP vous pouvez réaliser ce même ordonnancement/exécution paralléle ?
Vous donnerez la structure du programme avec un commentaire pour indiquer le travail de la tâche sous forme de commentaire : `// Travail ti`.