Faculté des Sciences & Techniques — Université de Limoges

*Master 1ère année*

*Parallélisme & Applications*

TD n°3

*Programmation OpenMP — la directive « task »*

*La directive « task »*

- la directive « task » crée une tâche de manière **explicite** ;
- toutes les threads se **partagent le travail disponible** parmi l'ensemble des tâches créées ;
- la directive « taskwait » **garantie** que toutes les tâches enfants crées dans la même tâche courante **ont fini**.

**1 –** Comparez les deux programmes suivants :

```
 1 #include <omp.h>
 2 #include <stdio.h>
 3
 4 main()
 5 {
 6    int i;
 7    #pragma omp parallel private(i)
 8    {    for(i=0; i<10; i++)
 9       {       printf("Task %d: %i\n",
10               omp_get_thread_num(),i);
11       }
12    }
13 }
```

```
 1 #include <omp.h>
 2 #include <stdio.h>
 3
 4 main()
 5 {
 6    int i;
 7    #pragma omp parallel private(i)
 8    {    for(i=0; i<10; i++)
 9       #pragma omp task
10       {       printf("Task %d: %i\n",
11               omp_get_thread_num(),i);
12       }
13    }
14 }
```

Combien de threads vont être utilisées et quel va être leur travail ?

**2 –** Que va produire le code suivant :

```
 1 #include <stdlib.h>
 2 #include <stdio.h>
 3 #include <omp.h>
 4
 5 int main()
 6 {
 7    #pragma omp parallel sections
 8    {
 9       #pragma omp section
10       {    int i;
11          for (i=0;i<4;i++)
12          {
13             #pragma omp task
14             printf("Task %d\n",omp_get_thread_num());
15          }
16       }
17       #pragma omp section
18       {    int i;
19          for (i=0;i<4;i++)
20            {
21             #pragma omp task
22             printf("Task %d\n",omp_get_thread_num());
23          }
24       }
25    }
26 }
```

**3 –** Décrivez le fonctionnement du programme suivant :

```
 1 #include <stdio.h>
 2 int work( int i )
 3 {
 4    if ( i > 0 )
 5    {
 6       #pragma omp parallel
 7       {
 8          #pragma omp task
 9          {
10             work( i-1 );
11          }
12          #pragma omp task
13          {
14             work( i-1 );
15          }
16          #pragma omp taskwait
17          printf( "Completed %i\n", i );
18       }
19    }
20 }
21 void main()
22 {
23    work( 3 );
24 }
```

**4 –** Essayez de **paralléliser** ces morceaux de code ou expliquez **pourquoi** ce n'est pas possible :

a.

```
1 for (i = 0; i < sqrt(x); i++)
2 {
3     a[i] = 2.3 * i;
4     if (i < 10)
5         b[i] = a[i];
6 }
```

b.

```
1 flag = 0;
2 for (i = 0; i < n && !flag; i++)
3 {
4          a[i] = 2.3 * i;
5          if (a[i] < b[i])
6          flag = 1;
7 }
```