Използване на OpenMP - част 5. Sections, flush, threadprivate.

Курс „Паралелно програмиране“

ИНСТИТУТ *за* СЪВРЕМЕННИ ФИЗИЧЕСКИ ИЗСЛЕДВАНИЯ

Стоян Мишев

План

Sections. Определение.

“*The sections construct is a noniterative worksharing construct that contains a set of structured blocks* ***that are to be distributed among and executed by the threads*** *in a team.* ***Each structured block is executed <u>once</u> by one of the threads*** *in the team in the context of its implicit task.*”

Sections

```c
#pragma omp parallel sections
{
    #pragma omp section
    {
        printf ("id = %d, \n", omp_get_thread_num());
    }

    #pragma omp section
    {
        printf ("id = %d, \n", omp_get_thread_num());
    }
}
```

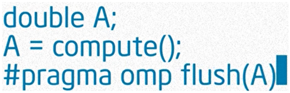
```
id = 0,
id = 1,
```

Flush. Определение.

При изпълнението на `flush`, дадена нишка има “consistent” достъп до мястото в паметта, описано от `flush set`.

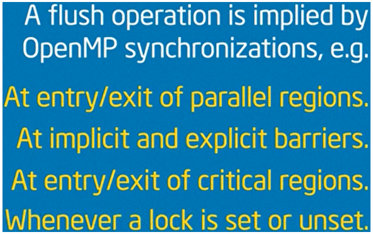
Flush. Определение.

При изпълнението на `flush`, дадена нишка има “consistent” достъп до мястото в паметта, описано от `flush set`.



```
double A;
A = compute();
#pragma omp flush(A)
```

Flush. Подразбиране



A flush operation is implied by OpenMP synchronizations, e.g.

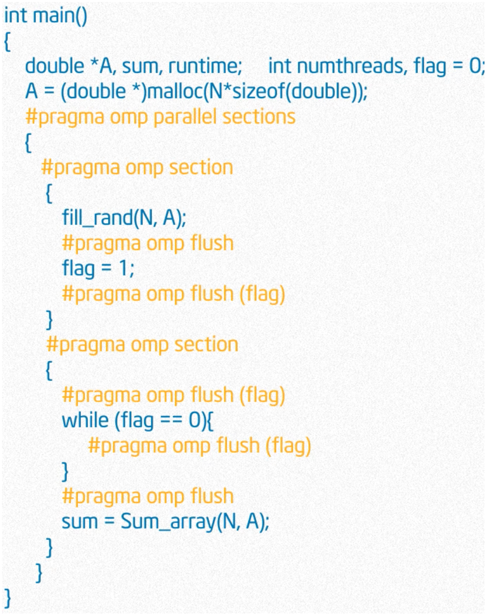
At entry/exit of parallel regions.

At implicit and explicit barriers.

At entry/exit of critical regions.

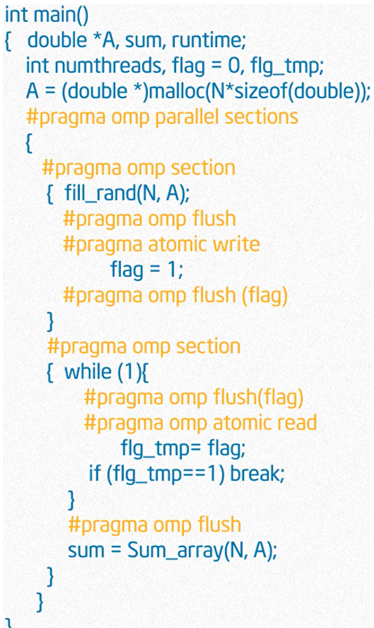
Whenever a lock is set or unset.

Producer-consumer



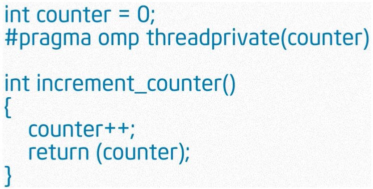
```
int main()
{
   double *A, sum, runtime;     int numthreads, flag = 0;
   A = (double *)malloc(N*sizeof(double));
   #pragma omp parallel sections
   {
     #pragma omp section
      {
        fill_rand(N, A);
        #pragma omp flush
        flag = 1;
        #pragma omp flush (flag)
      }
      #pragma omp section
      {
        #pragma omp flush (flag)
        while (flag == 0){
            #pragma omp flush (flag)
        }
        #pragma omp flush
        sum = Sum_array(N, A);
      }
    }
}
```

Producer-consumer



```
int main()
{   double *A, sum, runtime;
    int numthreads, flag = 0, flg_tmp;
    A = (double *)malloc(N*sizeof(double));
    #pragma omp parallel sections
    {
      #pragma omp section
      {  fill_rand(N, A);
         #pragma omp flush
         #pragma atomic write
                flag = 1;
         #pragma omp flush (flag)
      }
      #pragma omp section
      {  while (1){
            #pragma omp flush(flag)
            #pragma omp atomic read
                  flg_tmp= flag;
            if (flg_tmp==1) break;
         }
         #pragma omp flush
         sum = Sum_array(N, A);
      }
    }
}
```

threadprivate



```
int counter = 0;
#pragma omp threadprivate(counter)

int increment_counter()
{
   counter++;
   return (counter);
}
```

Домашна работа

от $Introduction\ to\ OpenMP_18Module10$