Използване на OpenMP - част 4. Tasks.

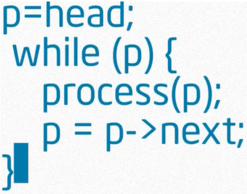
Курс „Паралелно програмиране“

ИНСТИТУТ *за* СЪВРЕМЕННИ
ФИЗИЧЕСКИ ИЗСЛЕДВАНИЯ

Стоян Мишев

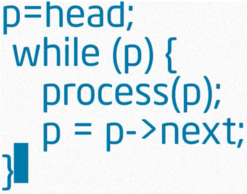
План

Свързан списък. Задача.



```
p=head;
 while (p) {
   process(p);
   p = p->next;
}
```

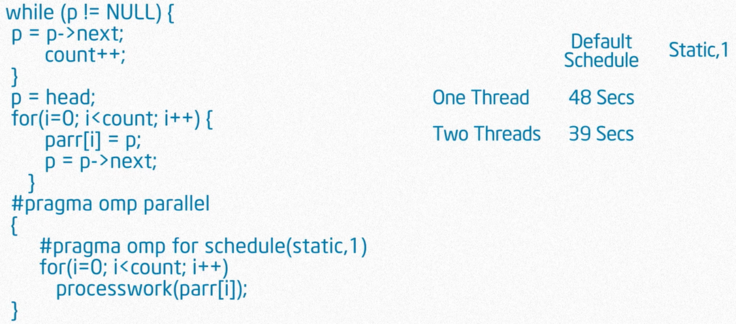
Свързан списък. Задача.



```
p=head;
 while (p) {
   process(p);
   p = p->next;
}
```

При `for` броят на интерациите е известен по време на компилиране. При `while` той зависи от условие, което се удовлетворява в процеса на изпълнение на програмата.

Свързан списък. Решение 1.



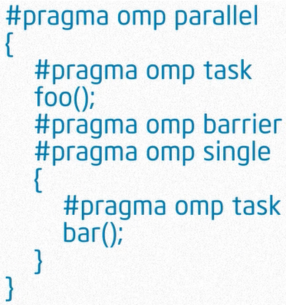
```
while (p != NULL) {
 p = p->next;
      count++;
 }
 p = head;
 for(i=0; i<count; i++) {
      parr[i] = p;
      p = p->next;
   }
 #pragma omp parallel
 {
      #pragma omp for schedule(static,1)
      for(i=0; i<count; i++)
         processwork(parr[i]);
 }
```

	Default Schedule	Static,1
One Thread	48 Secs	
Two Threads	39 Secs	

- цикъл 1. установяне на броя елементи
- цикъл 2. записване на указателите в масив
- цикъл 3. `pragma omp parallel for`

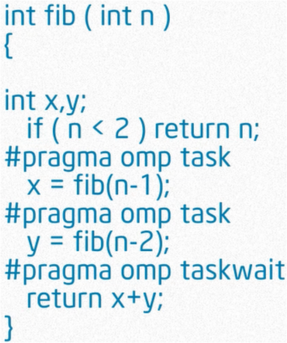
Задачи в OpenMP(tasks)

Обособени единици за изпълнение със собствен код и данни, които се “вземат” от различни нишки чрез планировчик (scheduler - internal control variables). Въведени в OpenMP 3.0.



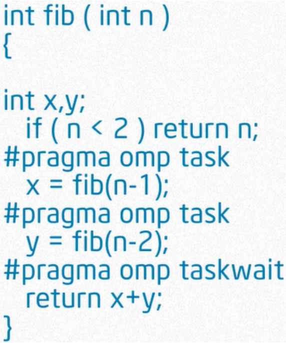
```
#pragma omp parallel
{
    #pragma omp task
    foo();
    #pragma omp barrier
    #pragma omp single
    {
        #pragma omp task
        bar();
    }
}
```

Задачи в OpenMP. Fibonacci.

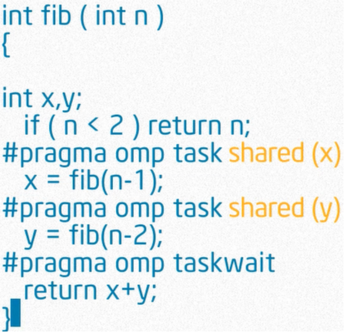


```
int fib ( int n )
{

int x,y;
   if ( n < 2 ) return n;
#pragma omp task
   x = fib(n-1);
#pragma omp task
   y = fib(n-2);
#pragma omp taskwait
   return x+y;
}
```

Задачи в OpenMP. Fibonacci.

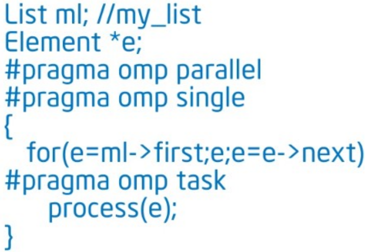


```
int fib ( int n )
{

int x,y;
   if ( n < 2 ) return n;
#pragma omp task
   x = fib(n-1);
#pragma omp task
   y = fib(n-2);
#pragma omp taskwait
   return x+y;
}
```

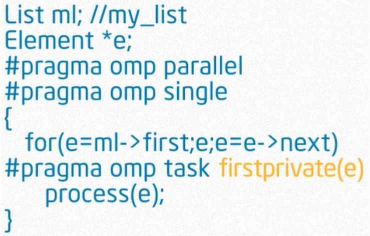


```
int fib ( int n )
{

int x,y;
   if ( n < 2 ) return n;
#pragma omp task shared (x)
   x = fib(n-1);
#pragma omp task shared (y)
   y = fib(n-2);
#pragma omp taskwait
   return x+y;
}
```

Задачи в OpenMP. Списък.

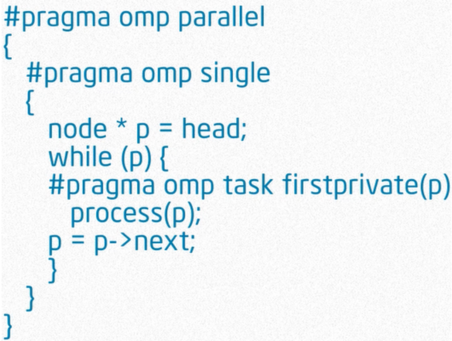


```
List ml; //my_list
Element *e;
#pragma omp parallel
#pragma omp single
{
  for(e=ml->first;e;e=e->next)
#pragma omp task
     process(e);
}
```

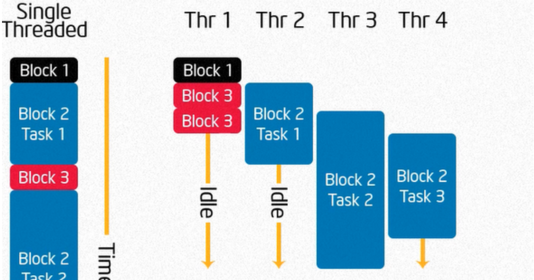


```
List ml; //my_list
Element *e;
#pragma omp parallel
#pragma omp single
{
  for(e=ml->first;e;e=e->next)
#pragma omp task firstprivate(e)
     process(e);
}
```

Задачи в OpenMP. Списък.



```
#pragma omp parallel
{
  #pragma omp single
  {
    node * p = head;
    while (p) {
    #pragma omp task firstprivate(p)
      process(p);
    p = p->next;
    }
  }
}
```



Разлика между task и section

“OpenMP’s section can be thought of as a static number of blocks of code to be executed in parallel. By static, I mean the number of blocks is to be fixed and known at compile time. On the other hand, task can be thought of as a way to break down into a dynamic number of blocks to be executed in parallel. By dynamic, I mean the number of blocks is not determined at compile time and depends on runtime.”

`https://medium.com/@techhara/openmp-section-vs-task-495b479ef317`

Домашна работа

от *Introduction to OpenMP 14 Module 8* до *Introduction to OpenMP 17 Discussion 7*