Общие механизмы OpenMP. Зависимость по данным

Data dependency
`for(i=1; i<8; i++)` `  a[i] = c*a[i-1]; ← зависимость есть`
`for(i=1; i<9; i+=2)` `  a[i] = c*a[i-1]; ← зависимости нет`

- *Утверждение 1*

  Только те переменные, в которые происходит запись на одной итерации и чтение их значения на другой, создают зависимость по данным.

- *Утверждение 2*

  Только разделяемые переменные могут создавать зависимость по данным.

- *Следствие*

  Если переменная не объявлена как приватная, она может оказаться разделяемой.

Общие механизмы OpenMP. Зависимость по данным

Function calls

```
double foo(double *a, double *b, int i) {
    return 0.345*(a[i] + b[2*i]*C); ← зависимость есть
}
```

```
double bar(double a, double b) {
    return 0.345*(a + b*C); ← зависимости нет
}
```

- Функцию необходимо сделать независимой от внешних данных, кроме как от **значения** параметров.
- В функции так же не должно быть статических переменных (**static**).

Общие механизмы OpenMP. Зависимость по данным

Function calls

```
#pragma omp for default(shared) private(i)
  for(i = 0; i < 10; i++)
    val[i] = func(i);
```

```
double func(int i) {
  double X;
  X=sqrt(2)*i;
  return X;
}
```

```
double func(int i) {
  static double X;
  X=sqrt(2)*i;
  return X;
}
```

```
double func(int i) {
  static double X;
  #pragma omp threadprivate(X)
  X=sqrt(2)*i;
  return X;
}
```

Общие механизмы OpenMP. Зависимость по данным

Nested loops

```
for(k=0; k<N; k++) ← зависимость есть
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            a[i][j] += b[i][k]*c[k,j];
```

```
for(i=0; i<N; i++)
    for(j=0; j<N; j++)
        for(k=0; k<N; k++) ← зависимости нет
            a[i][j] += b[i][k]*c[k,j];
```

- Циклы, в которых есть **выход по условию,** не должны подвергаться распараллеливанию

Общие механизмы OpenMP. Средства синхронизации

#pragma omp critical [(name)]

```
#pragma omp for private(i) shared(a,xmax)
  for(i=0; i<N; i++) {
    #pragma omp critical (xmax)
    {
      if(a[i]>xmax)
        xmax = a[i];
    }
  }
```

- Правилом хорошего тона считается, если критическая секция содержит обращения только к одному разделяемому ресурсу.
- Все безымянные секции рассматриваются как одна (очень большая), и если не давать им явно имена, то только в одной из этих секций в один момент времени будет один поток - остальные будут ждать.

Общие механизмы OpenMP. Средства синхронизации

#pragma omp barrier

```
#pragma omp parallel
{
  #pragma omp atomic
      value++;
  #pragma omp barrier
  #pragma omp critical (cout)
  {
    std::cout << value << std::endl;
  }
}
```

Общие механизмы OpenMP. Средства синхронизации

#pragma omp ordered

```
#pragma omp parallel private(myid)
{
  myid = omp_get_thred_num( );
  #pragma omp for private(i)
    for(i=0; i<8; i++)
      #pragma omp ordered
      printf("T%d: %d\n", myid, i);
}
```

```
T0: 0
T0: 1
T0: 2
T0: 3
T1: 4
 …
```

Общие механизмы OpenMP. Средства синхронизации

```
#pragma omp flush(var1[, var2, ...])
```

- Осуществляет немедленный сброс значений разделяемых переменных в память.
- Таким образом гарантируется, что во всех потоках значение переменной будет одинаковое.
- Неявно присутствует в директивах: `barrier`, начале и конце критических секций, параллельных циклов, параллельных областей, `single` секций.
- С ее помощью можно посылать сигналы потоком используя переменную как семафор. Когда поток видит, что значение разделяемой переменной изменилось, это говорит о том, что произошло событие и можно продолжить выполнение программы далее.