Open Multi-Processing

Перов Максим
кафедра РЭПИ, МФТИ(ГУ)

Директива firstprivate

```
int i = 0;
...
#pragma omp parallel for firstprivate(i)
        for (i; i < K; i++) // параллельная печать чисел 0..K
                printf("i: %i\n", i); // с теоретическим ускорением N, где N - число потоков
```

firstprivate - работает как и **private**, но инициализирует переменную значением, которое было получено в предыдущем последовательном коде.

Директива lastprivate

```
int i;
...
#pragma omp parallel for lastprivate(i)
        for (i = 0; i < K; i++) // параллельная печать чисел 0..K
                printf("i: %i\n", i); // с теоретическим ускорением N, где N - число потоков
```

lastprivate - работает как и **private**, но значение переменной, вычисленное на последней итерации цикла, сохраняется.

Директива default

```
int i, j, g;
...
#pragma omp parallel default(private) shared(g)
{
      // i и j будут частными по умолчанию, а g будет общей
}
```

```
default(shared | private)
```

default - задаёт область видимости переменных внутри региона по умолчанию.

Директива schedule

schedule(тип, *размер блока*)

- **static** - итерации равномерно распределяются по потокам; (*используется по умолчанию*)
- **dynamic** - работа распределяются блоками заданного размера между потоками;
- **guided** - как и **dynamic**, но размер блока может изменяться динамически в зависимости от того, сколько итераций осталось. Размер уменьшается вплоть до указанного значения.

Директива if

```
int i;
...
#pragma omp parallel for if(K > 2000)
        for (i = 0; i < K; i++) // параллельная печать чисел 0..K
                printf("i: %i\n", i); // с теоретическим ускорением N, где N - число потоков
```

if(условное выражение)

if - позволяет определять необходимость параллельного исполнения.

Директива reduction

```
int i, sum, K;
...
#pragma omp parallel for private(i) shared(K) reduction(+:res)
        for (i = 0; i < K; i++) // параллельное суммирование чисел 0..K
                res += i; // с теоретическим ускорением N, где N - число потоков
```

reduction(оператор : переменная)

Операторы для **reduction** в C\C++: **+**, **-** , *, **&**, **^**, **|**, **&&**, **||**, **min**, **max**

Задание на семинаре

Разработать openMP-программу, которая суммирует ряд:

$$\sum 1/n!$$

суммирование осуществлять от 0 до N, где N задается через аргументы программы. На выходе программы **результат суммирования**, **ускорение** и **эффективность**.

Отсчет времени в Linux

```
#include <sys/time.h> // для функции gettimeofday и вспомогательных структур

struct timeval tv1, tv2, dtv;
struct timezone tz;

void time_start(); // функция начала отсчёта времени
long time_end(); // функция завершения отсчёта времени

int main(int argc, char **argv) {
        time_start();
        …
        printf("Время работы программы: %ld\n мс.", time_end());
        return 0x00;
}
```

Отсчет времени в Linux

```
#include <sys/time.h> // для функции gettimeofday и вспомогательных структур
struct timeval tv1, tv2, dtv;
struct timezone tz;
void time_start() { gettimeofday(&tv1, &tz); } // функция начала отсчёта времени

long time_end() { // функция завершения отсчёта времени
        gettimeofday(&tv2, &tz);
        dtv.tv_sec = tv2.tv_sec - tv1.tv_sec;
        dtv.tv_usec = tv2.tv_usec - tv1.tv_usec;
        if ( dtv.tv_usec < 0 ) {
                dtv.tv_sec--;
                dtv.tv_usec += 1000000;
        }
        return dtv.tv_sec*1000 + dtv.tv_usec/1000;
}
```