Технологии параллельного программирования

Модуль 1. Технология OpenMP

Семинар 7

Скомпилировать и выполнить программный код на языке С++. Полученные результаты представить в отчете (скринами) преподавателю, сделать краткие выводы.

**Задание 1. Использование директивы For и функции rand для заполнения матрицы в параллельной секции**

Программа инициализирует двумерный массив arr рандомными значениями, делая все действия параллельно. С помощью получения текущего времени и прибавления номера потока мы инициализируем функцией srand генератор случайных чисел rand и получаем действительно случайный результат.

Показать также результат работы программы без инициализации.

#include <iostream>

#include "omp.h"

#include <locale>

#include <ctime>

using namespace std;

int main()

{

    double start, end;

    const int W = 10;

    int arr[W][W];

    start = omp\_get\_wtime();

#pragma omp parallel

    {

       srand(time(0) + omp\_get\_thread\_num());

#pragma omp for schedule(dynamic, 6)

        for (int i = 0; i < W; ++i)

            for (int j = 0; j < W; ++j)

                arr[i][j] = rand() % 50;

    }

    end = omp\_get\_wtime();

    for (int i = 0; i < W; i++)

    {

        for (int j = 0; j < W; j++)

        {

            cout << arr[i][j];

            cout << ' ';

        }

        cout << "\n";

    }

    printf("Compute Time: %f seconds\n", end - start);

}

**Задание 2. Директива Barrier**

#include <stdio.h>

#include <omp.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

#pragma omp parallel

{

printf("Сообщение 1\n");

printf("Сообщение 2\n");

#pragma omp barrier

printf("Сообщение 3\n");

}

}

**Задание 3. Директива Critical**

#include <stdio.h>

#include <omp.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

#pragma omp parallel

{

#pragma omp critical

{

n = omp\_get\_thread\_num();

printf("Поток %d\n", n);

}

}

}

**Задание 4. Директива Atomic**

#include <stdio.h>

#include <omp.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

int count = 0;

#pragma omp parallel

{

#pragma omp atomic

count++;

}

printf("Число потоков: %d\n", count);

}