Технологии параллельного программирования

Модуль 1. Введение. Технология OpenMP

Семинар 4

**Общие задания**

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

**Задание 1. Применение функции omp\_set\_num\_threads и опции num\_threads**

#include <stdio.h>

#include <omp.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

omp\_set\_num\_threads(2);

#pragma omp parallel num\_threads(3)

{

printf("Параллельная область 1\n");

}

#pragma omp parallel

{

printf("Параллельная область 2\n");

}

return 0;

}

**Задание 2. Вложенные параллельные области (nested).**

#include <stdio.h>

#include <omp.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

omp\_set\_num\_threads(3);

omp\_set\_nested(1); //разрешаем использование вложенных параллельных областей - устанавливаем переменную окружения

#pragma omp parallel private(n)

{

n = omp\_get\_thread\_num(); // определение номера потока во внешней параллельной секции

printf("Внешняя параллельная секция 1, поток n=%d \n", n);

#pragma omp parallel

{

printf("Вложенная параллельная секция 1, поток %d - %d\n", n, omp\_get\_thread\_num());

}}

omp\_set\_nested(0); //запрещаем использование вложенных параллельных областей - сбрасываем переменную окружения

#pragma omp parallel private(n)

{

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

printf("Внешняя параллельная секция 2, поток n=%d \n", n);

#pragma omp parallel

{

printf("Вложенная параллельная секция 2, поток %d - %d\n", n,

omp\_get\_thread\_num());

}}

return 0;

}

**Задание 3. Директива single, опция copyprivate.**

#include <iostream>

#include <stdio.h>

#include <omp.h>

int main()

{

int num;

setlocale(LC\_ALL, "Russian");

#pragma omp parallel num\_threads(4) private(num)

{

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

printf("Перед директивой single num=%d \n", num);

#pragma omp barrier

#pragma omp single copyprivate(num)

{

printf("Введите целое число : ");

scanf("%d", &num);

                        printf("Введено число : num=%d \n", num);

}

printf("После директивы single num=%d \n", num);

}

}

**Задание 4. Директива single, опция nowait.**

#include <iostream>

#include <stdio.h>

#include <omp.h>

int main()

{

setlocale(LC\_ALL, "Russian");

double k = 0;

#pragma omp parallel num\_threads(4) firstprivate(k)

{

printf("Перед директивой single без nowait \n");

#pragma omp single

{

// Данный цикл добавлен для того, чтобы поток выполнил какую-нибудь работу

for (int i = 0; i < 100000; i++)

{

k += (double)i / (i + 1);

}

printf("В директиве single \n");

}

printf("После директивы single без nowait. Это сообщение никогда не будет раньше предыдущих, напечатанных в директиве single. k = %f \n",k);

#pragma omp barrier // Данная директива синхронизирует потоки

printf("Перед директивой single c nowait \n");

#pragma omp single nowait

{

// Данный цикл добавлен для того, чтобы поток выполнил какую-нибудь работу

for (int i = 0; i < 100000; i++)

{

k += (double)i / (i + 1);

}

printf("В директиве single \n");

}

printf("После директивы single c nowait. Это сообщение может быть раньше предыдущих, напечатанных в директиве single. k = %f \n", k);

}

return 0;

}

**Задания для самостоятельного выполнения**

1. Определите, какое максимальное количество нитей позволяет породить для выполнения параллельных областей программы ваша система.
2. При помощи трёх уровней вложенных параллельных областей породите 8 нитей (на каждом уровне параллельную область должны исполнять 2 нити). Посмотрите, как будет исполняться программа, если запретить вложенные параллельные области.