

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школы электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ
о лабораторной работе №1
по дисциплине «Технологии параллельного программирования»

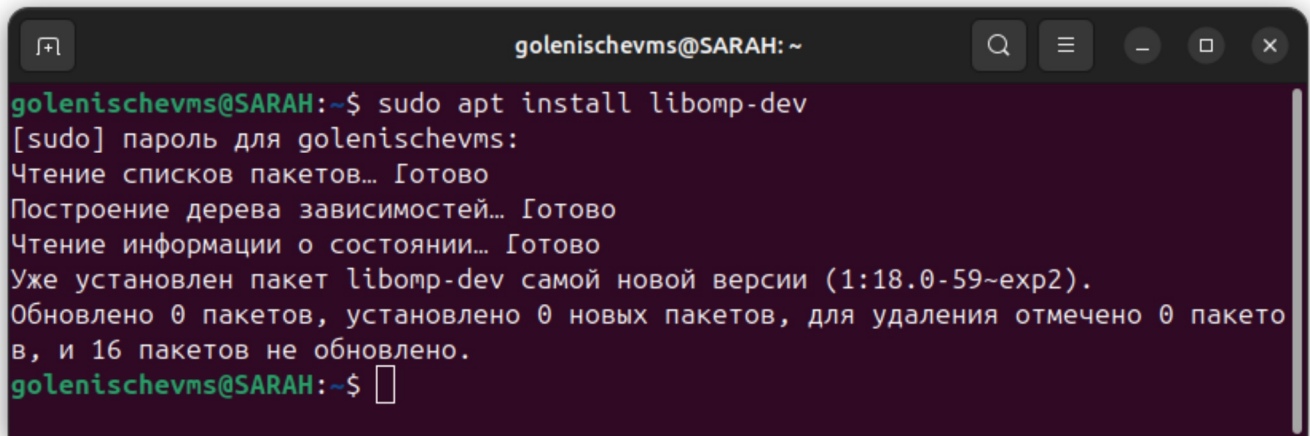
Выполнил:
студент группы КЭ-220
_____/Голенищев А. Б.
_____ 2024 г.

Отчет принял:
_____/Жулев А. Э.
_____ 2024 г.

Челябинск
2024

Задание 1. Создание проекта в среде MS Visual Studio с поддержкой OpenMP
IDE: Qt Creator.

Создали проект приложения на C++, система сборки qmake. Установили OpenMP в наш дистрибутив, рисунок 1. Открываем файл проекта с расширением *.pro, подключаем поддержку OpenMP в несколько строк, листнинг 1.



```
golenischevms@SARAH: ~  
golenischevms@SARAH:~$ sudo apt install libomp-dev  
[sudo] пароль для golenischevms:  
Чтение списков пакетов... Готово  
Построение дерева зависимостей... Готово  
Чтение информации о состоянии... Готово  
Уже установлен пакет libomp-dev самой новой версии (1:18.0-59~exp2).  
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов,  
и 16 пакетов не обновлено.  
golenischevms@SARAH:~$
```

Рисунок 1. Установка OpenMP в Ubuntu

```
TEMPLATE = app  
CONFIG += console c++17  
CONFIG -= app_bundle  
CONFIG -= qt  
  
# Флаги компилятора и линковки для OpenMP  
QMAKE_CXXFLAGS += -fopenmp  
QMAKE_LFLAGS += -fopenmp  
  
SOURCES += \  
main.cpp
```

Листнинг 1. Настройка файла проекта Qt для работы с OpenMP

Задание 2. Многопоточная программа «Hello World!»

Написали первую многопоточную программу, листинг 2. Результат ее работы представлен на рисунке 2. Процессор: Intel Core i7-13700K (16 ядер, 24 потока). В выводе количество выведенных «Hello World!» соответствует количеству потоков процессора.

```
#include <stdio.h>
#include <omp.h>
// Golenishchev Artem, KE-220 Task 2
int main() {
    // Параллельный регион
    #pragma omp parallel
    {
        printf("Hello, World!\n");
    }
    return 0;
}
```

Листинг 2. Код параллельной программы «Hello World!»

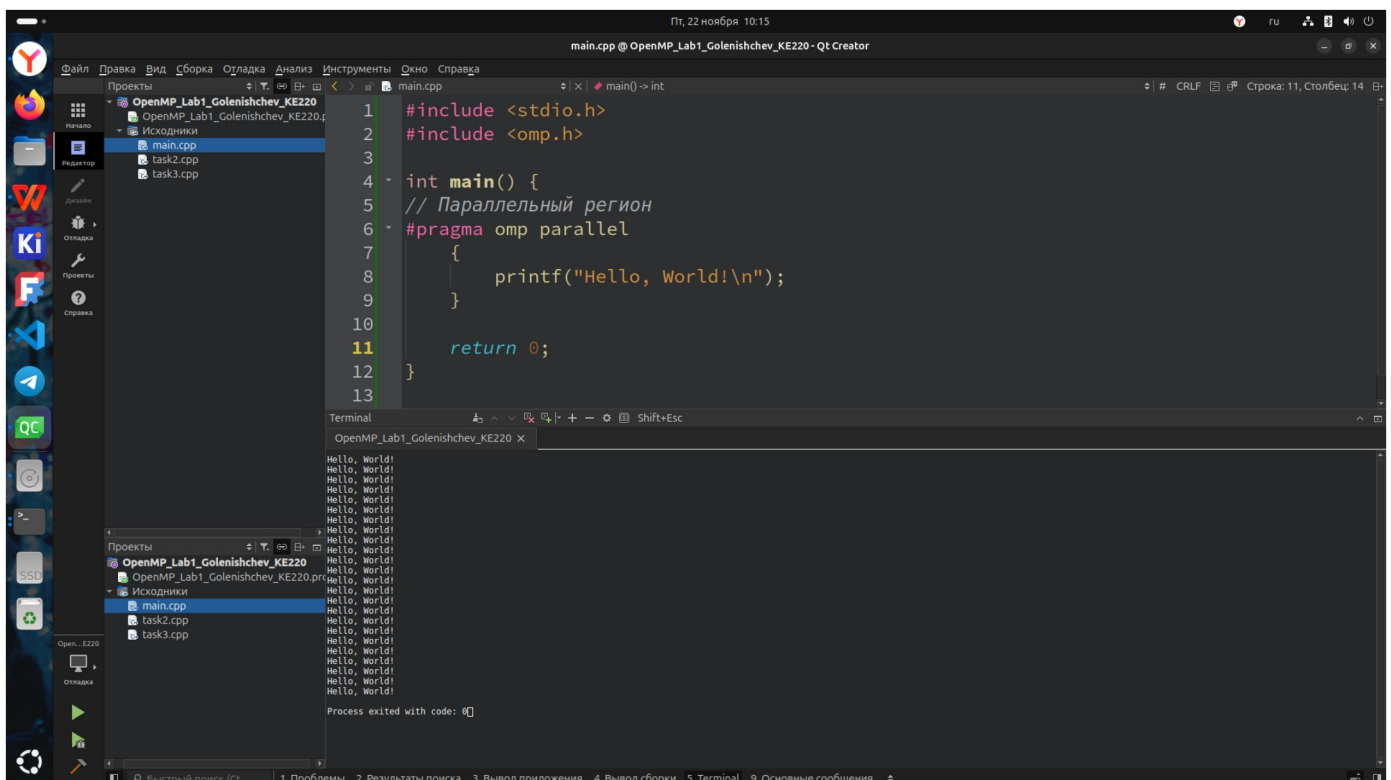


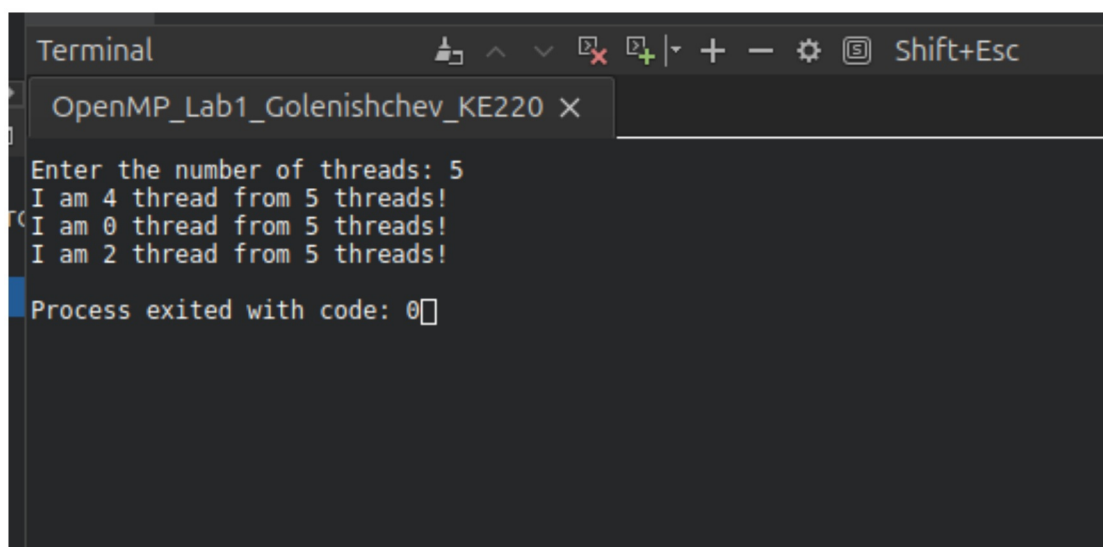
Рисунок 2. Работа первой многопоточной программы

Задание 3. Программа «I am!»

Написали программу, в которой создается k нитей, и каждая нить выводит на экран свой номер и общее количество нитей в параллельной области в формате, листинг 3. Представлен результат ее выполнения, рисунок 3. I am <thread> from <threads>!

```
#include <stdio.h>
#include <omp.h>
// Golenishchev Artem, KE-220 Task 3
int main() {
    int k;
    printf("Enter the number of threads: ");
    scanf("%d", &k);
    omp_set_num_threads(k);
    #pragma omp parallel
    {
        // Получаем идентификатор текущего потока (нити)
        int thread_id = omp_get_thread_num();
        // Получаем общее количество потоков
        int num_threads = omp_get_num_threads();
        // Проверяем, является ли номер нити четным
        if (thread_id % 2 == 0) {
            printf("I am %d thread from %d threads!\n", thread_id, num_threads);
        }
    }
    return 0;
}
```

Листинг 3. Программа вывода номеров четных нитей с OpenMP



```
Terminal
OpenMP_Lab1_Golenishchev_KE220 x
Enter the number of threads: 5
I am 4 thread from 5 threads!
I am 0 thread from 5 threads!
I am 2 thread from 5 threads!
Process exited with code: 0
```

Рисунок 3. Результат определения четных нитей из введенного общего количества

Ответы на вопросы к лабораторной работе:

1. Что такое OpenMP? Какие модели он реализует? Опишите модели и их связь.

OpenMP – это стандарт интерфейса для многопоточного программирования над общей памятью и набор средств компилируемых языков программирования C++ и Fortran. Модели:

- 1) Модель программирования в общей памяти. Параллельно приложение состоит из нескольких процессов, выполняющихся одновременно. Процессы разделяют общую память и обмены между процессами осуществляются чтением/записью данных в общей памяти.
- 2) Модель FORK-JOIN заключается в том, что программа сама является полновесным процессом, но она может запускать легковесные процессы (нити) в ходе выполнения, которым выделяется собственная память (сегмент стека) приложения. Процесс приложения - главная нить.

Данные модели между собой непосредственно не связаны, предназначены для разных целей.

2. В каких языках реализован этот стандарт? Из каких частей состоит реализация в Visual Studio? (вместо VS ответу про Qt)

Стандарт имеет реализацию на C++ и Fortran. Проект состоит из компилятора с поддержкой OpenMP, библиотеки и директив.

3. Какие существуют варианты задания количества нитей в параллельном регионе?

Сколько нитей будет создано, если указаны оба варианта с разными значениями?

Что конкретно делает функция `omp_set_num_threads()`?

Существует два варианта задания количества нитей в параллельном регионе в OpenMP:

Функция `omp_set_num_threads()`. Задаёт количество потоков в предстоящих параллельных регионах, если не переопределяется предложением `num_threads`.

Директива `#pragma omp parallel num_threads(num_threads)`. Явно задаёт количество нитей, которые будут выполнять параллельную область. По умолчанию

выбирается последнее значение, установленное с помощью функции `omp_set_num_threads()`, или значение переменной `OMP_NUM_THREADS`.

Функция `omp_set_num_threads()` устанавливает глобальное количество нитей для всех параллельных регионов, передав ей целое число, соответствующее желаемому количеству.

4. Как идентифицируются нити в OpenMP ? Для чего это нужно ? Приведите содержательный пример. Совпадают ли эти идентификаторы с идентификаторами потоков в ОС?

Потоки в OpenMP идентифицируются нумерацией от 0 до N (их количества). Идентификация потоков нужна для того, чтобы мы могли как в примере задания 3 иметь половину четных, половину нечетных потоков - поскольку они нумеруются от нуля подряд. Операционная система будет нумеровать потоки в псевдослучайном порядке - никто не может гарантировать что половина ID - четные, половина - нечетные. Обеспечивается независимость от операционной системы и компилятора, а также данные номера используются в механизме Control Flow (управление потоками в зависимости от условий, циклов, событий). Идентификаторы с теми, что выдает потокам операционная система не совпадают.

5. Каков порядок вывода сообщений нитями? Всегда ли он одинаков? Чем определяется этот порядок?

Порядок вывода сообщений нитями в OpenMP не является гарантированно одинаковым при каждом запуске программы. Он определяется внутренними механизмами планировщика потоков, которые зависят от особенностей операционной системы, компилятора, аппаратной архитектуры и текущей загрузки системы. В параллельной среде потоки выполняются независимо и могут завершать свои задачи в разной последовательности. Для управления порядком вывода можно использовать механизмы синхронизации, такие как директивы `#pragma omp critical`, `#pragma omp ordered` или функции блокировки, чтобы обеспечить предсказуемую последовательность сообщений.

Выводы:

Изучили стандарт OpenMP для организации параллельного программирования на языке C++. Рассмотрены методы задания количества нитей в параллельных регионах, особенности идентификации нитей, а также поведение программы при выводе данных из разных потоков. Практическая реализация показала, как можно контролировать выполнение параллельных участков кода и распределение задач между нитями. Также было выявлено, что порядок вывода сообщений из нитей непредсказуем и может зависеть от множества факторов, включая планировщик потоков операционной системы.