|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| **«Национальный исследовательский ядерный университет «МИФИ»**  **(НИЯУ МИФИ)** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (№12) |

Лабораторная работа №2

«Реализация через OpenMP»

по дисциплине «Гибридные суперкомпьютерные технологии»

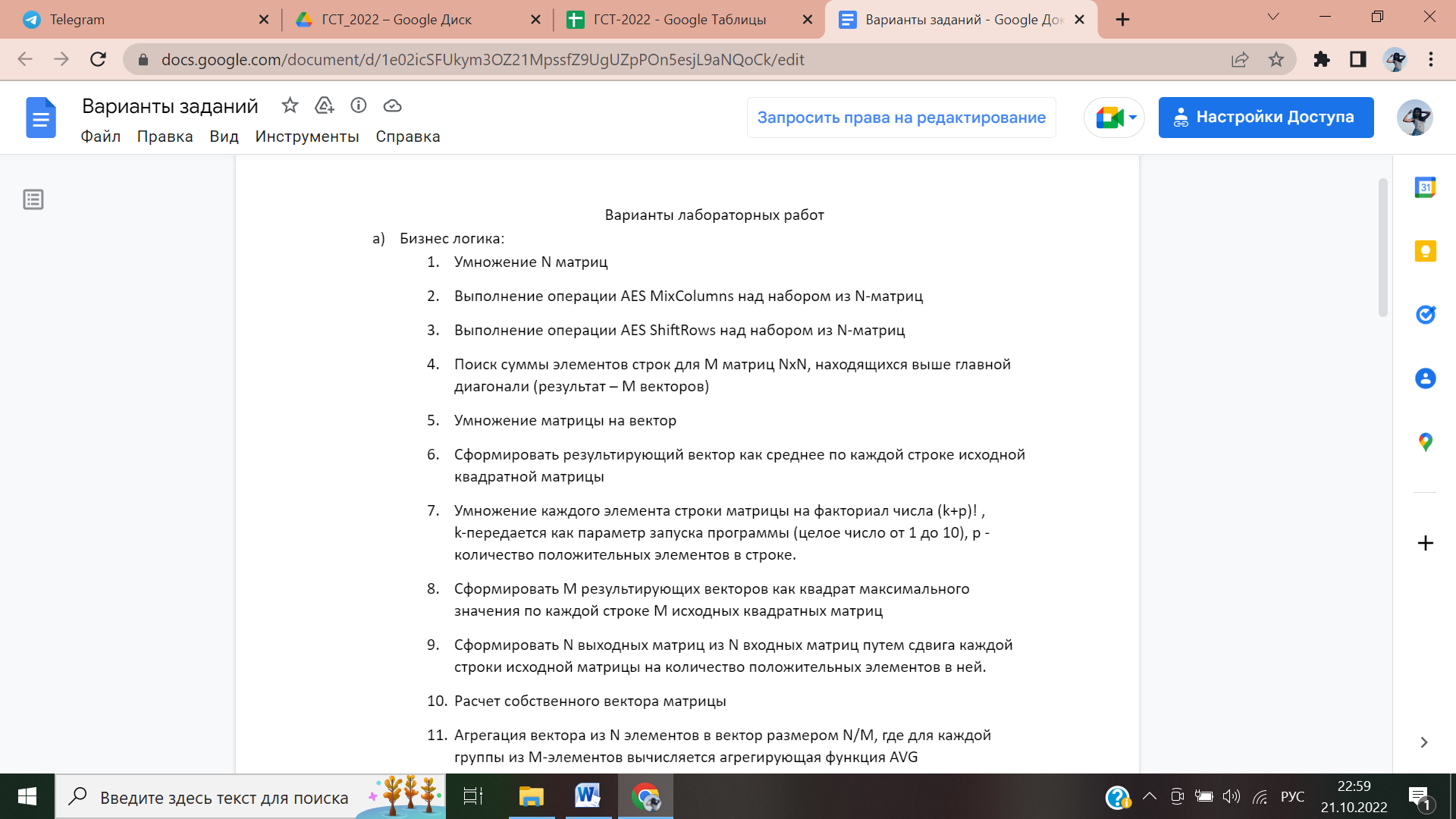
Работу выполнила студентка группы М21-502:

Нургазиева Д.Р.

Проверил: Синельников Д.М.

2022

**Цель:** изучить основы реализация программы через OpenMP. Выполнить задание по варианту 5:



Особенности исполнения в каждом из вариантов:

* Считывание данных происходит из файла (либо данные передаются по протоколу TCP)
* Данные генерируются утилитой, принимающей в качестве параметров размер данных для обработки в мегабайтах и имя файла (TCP хост-порт) куда будут выгружены данные
* Программа выполняет бизнес-логику и записывает результат в выходной файл (отправляет данные на порт возврата результатов программы-генератора по TCP, сохранение файла с результатами осуществляет программа-генератор)
* В конце файла с результатами сохраняется информация о времени выполнения вычислений и размере обработанных данных

# **Ход работы:**

Программа состоит из двух подпрограмм:

1. Generator.c – отвечает за генерацию данных;
2. Counter.c – выполняет умножение матрицы на вектор, подсчитывает время.

В первую очередь запускается generator.c и вводится размер файла и имя файла. Генерируется матрица с цифрами от 0 до 100. Количество строк и столбцов в зависимости от мегабайт высчитывается по формуле i

nt res = (dataInMb \* 1024) / sqrt(3\*dataInMb);

По сути, в среднем имеем 3 бита на одно число (пробел + само число в 90: случаев состоящее из 2х цифр). Сгенерированные данные грузятся файл

Данные генерируются в matrix.txt

Далее запускается counter.c и вводится размер файла и имя файла.

Происходит: 1. Выделение памяти под массив, матрицу. 2. перемножение матрицы на вектор, в рамках этого процесса подсчитывается время выполнения. Также происходит реализация через OpenMP. Приницип работы представлен на рисунке 1.

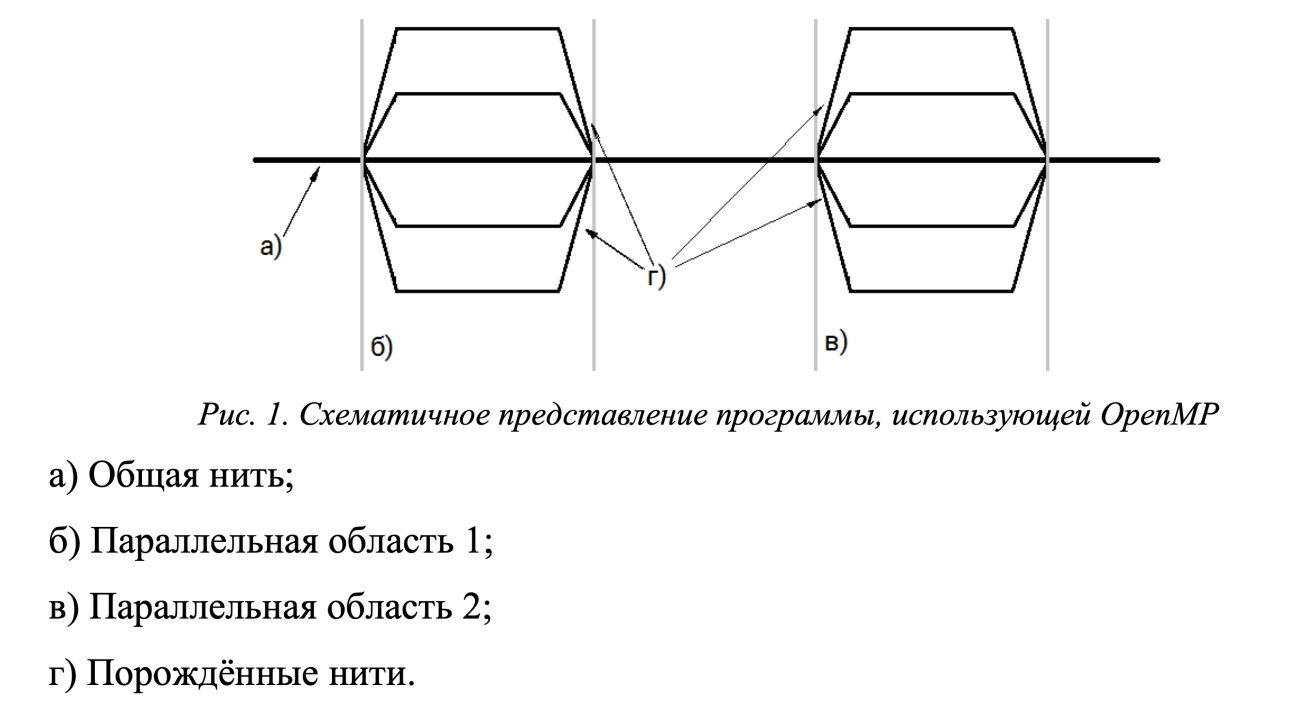
[](https://user-images.githubusercontent.com/72603507/193226323-79f2e306-d2b0-4ef0-b842-8b655f61a259.png)

Рисунок 1 - OpenMP

При использовании технологии OpenMP программа, как показано на рисунке 1, начинает свое выполнение как один процесс, называемый главной нитью. Главная нить выполняется последовательно до тех пор, пока не встретится первая параллельная область программы. Начало параллельной области определяется директивой процессору #pragma omp parallel, а её границы задаются фигурными скобками. При входе в параллельную область главная нить порождает некоторое число подчиненных ей нитей, которые вместе с ней образуют текущую группу нитей. Все операторы программы, находящиеся в параллельной конструкции, включая вызываемые изнутри нее функции, выполняются всеми нитями текущей группы параллельно до выхода из этой области. При выходе из параллельной области все порожденные на её входе нити сливаются с главной нитью, которая продолжает свое дальнейшее выполнение. При распараллеливании получается алгоритм ленточного вычисления. При данном подходе матрица “нарезается” на ленты, а те в свою очередь параллельно вычисляются.

**#pragma omp parallel for num\_threads(2) shared(matrix, arr, res\_matrix) private (i, j) reduction(+: data\_counter) schedule(static)**

**num\_threads**  - число потоков

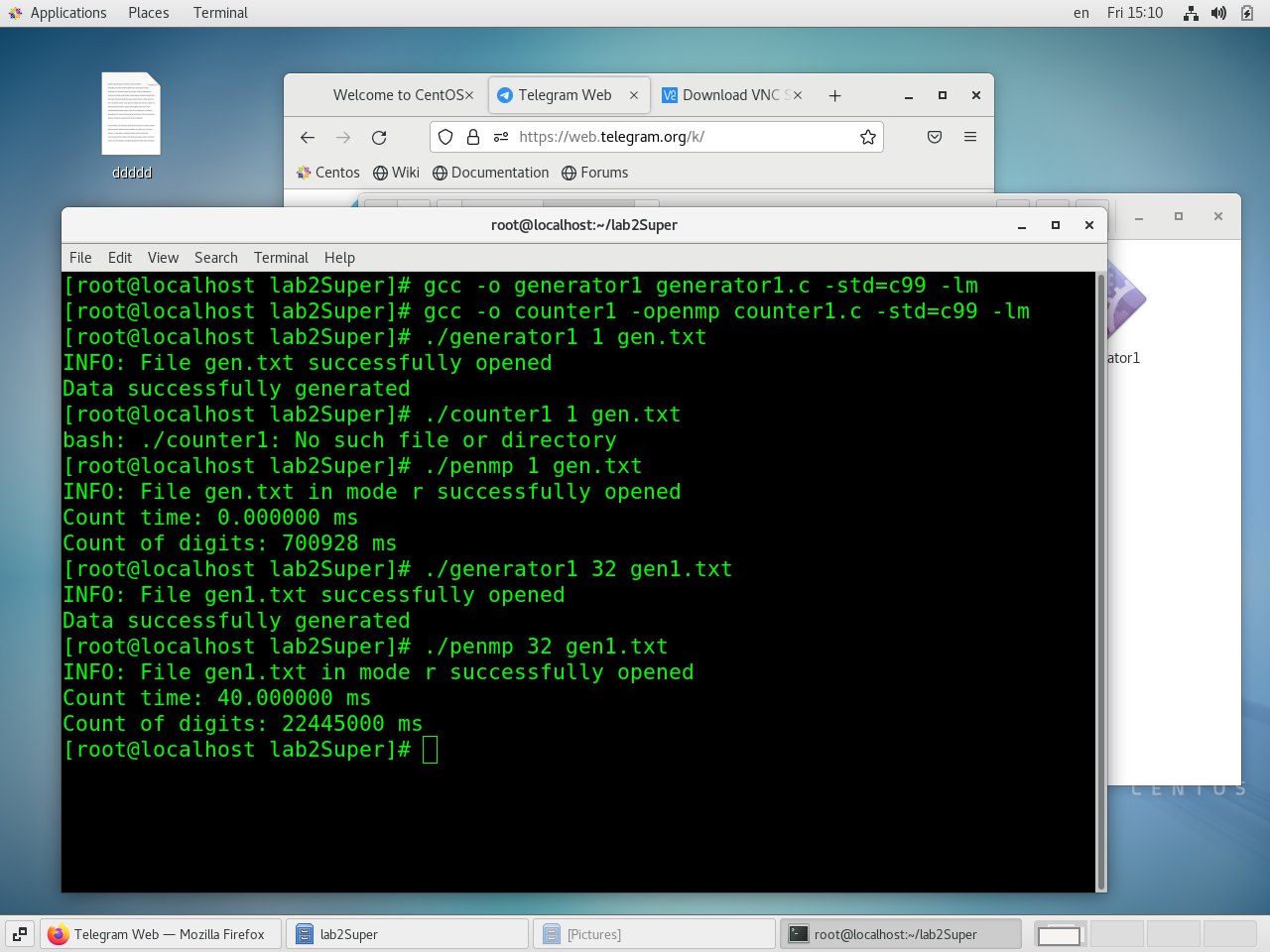
**shared**  - явным образом определяет список переменных, которые должны быть общими для всех нитей параллельной области.

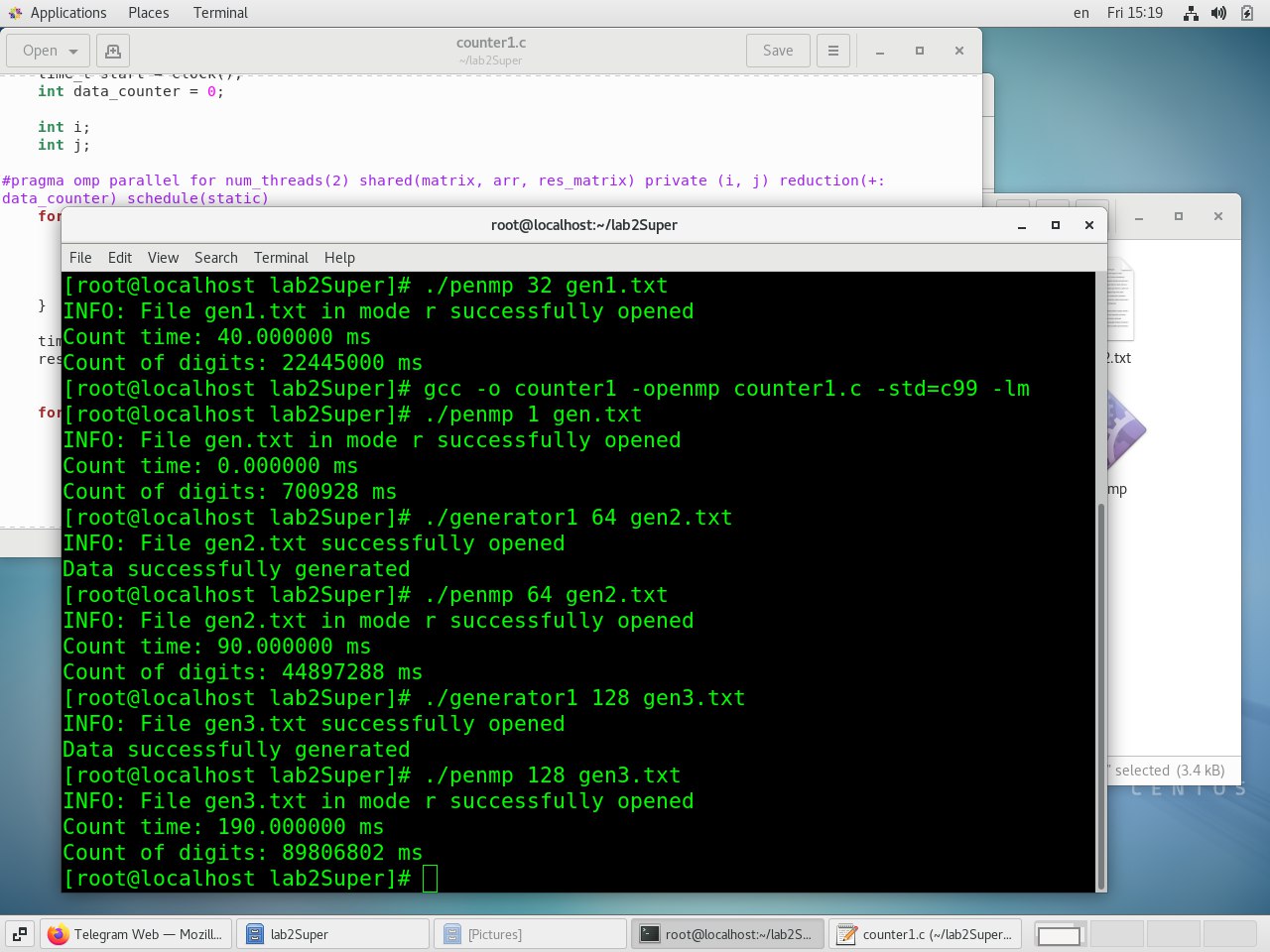
**private**  - Определяет список переменных, которые должны быть локальными для каждой нити.

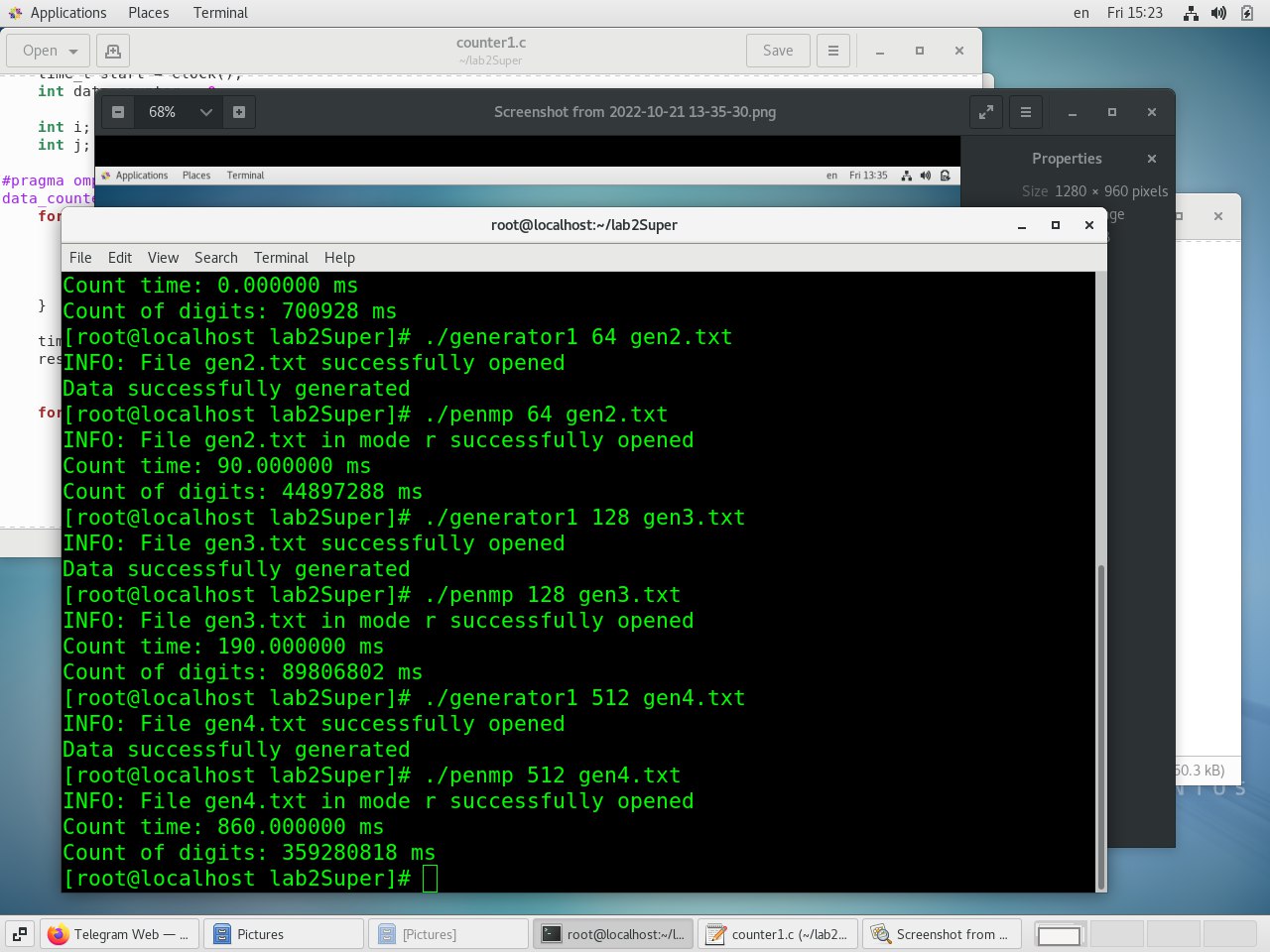
**reduction**  - Операция редукции производится над копиями переменных во всех нитях, и результат присваивается базовой переменной.

**shedule**  - Чтобы максимально сокращать время ожидания, необходимо распределить общую работу, чтобы все потоки поступали в барьер примерно в то же время.

**Выполнение программы:**

****

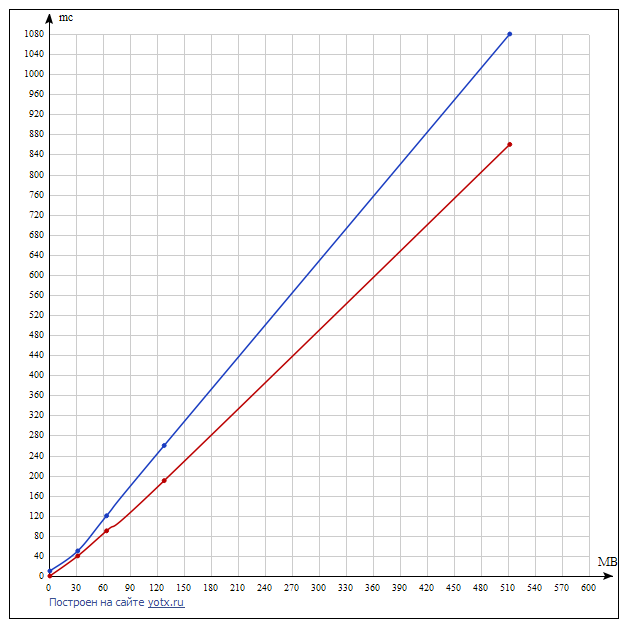
****

****

Сравнительная таблица параллельной реализации и реализации через OpenMP

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Время (мс)на 1мб | Время (мс)на 32мб | Время (мс)на 64мб | Время (мс)на 128мб | Время (мс)на 512мб |
| Параллельный | 10 | 50 | 120 | 260 | 1080 |
| OpenMP | 0 | 40 | 90 | 190 | 860 |

График времени выполнения:



**Вывод:** были изучены основы реализации OpenMP. Проанализировано время выполнения программы при разных подходах.