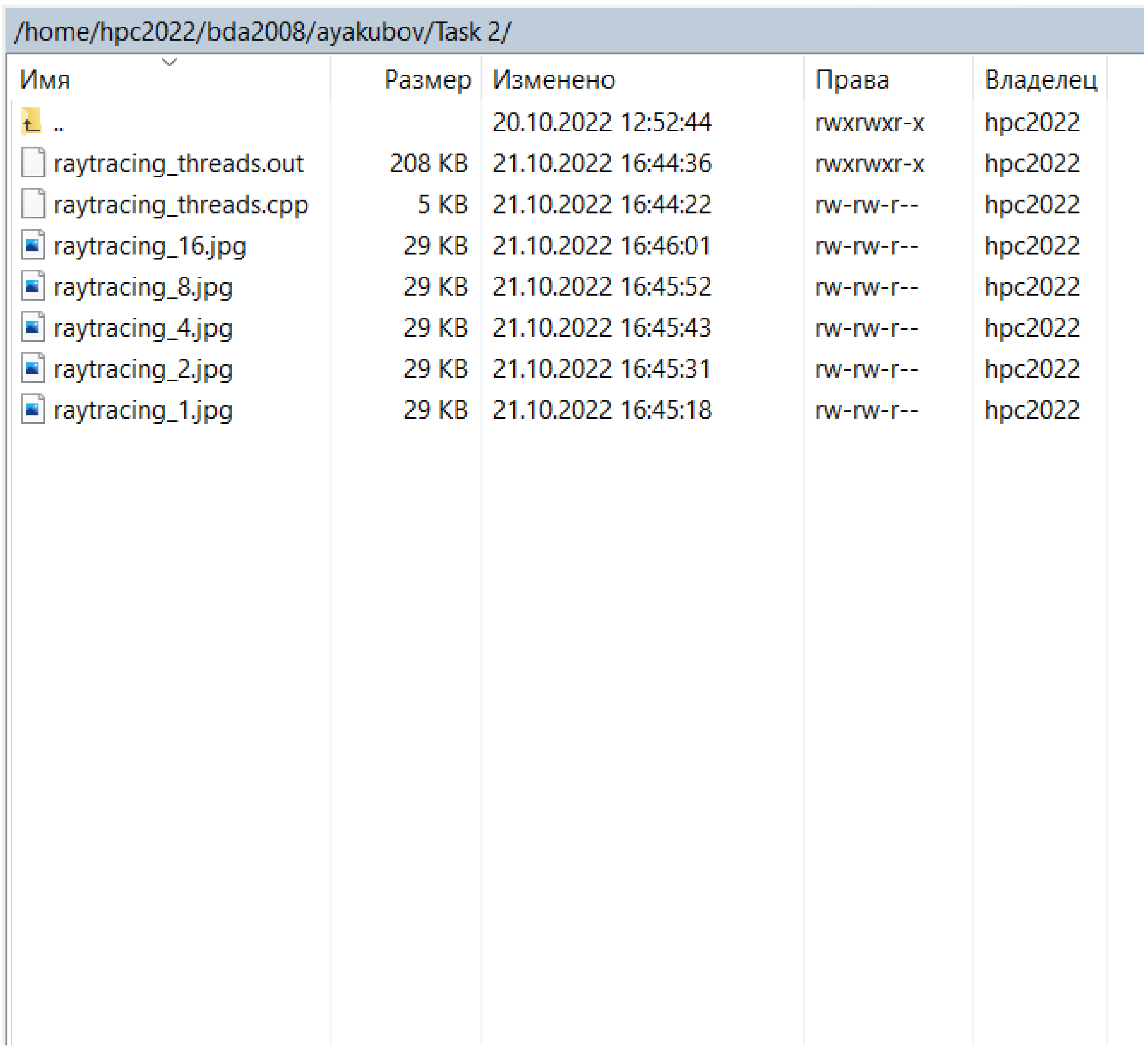
Amir Yakubov

BD-2008

Task 2

Step 1.

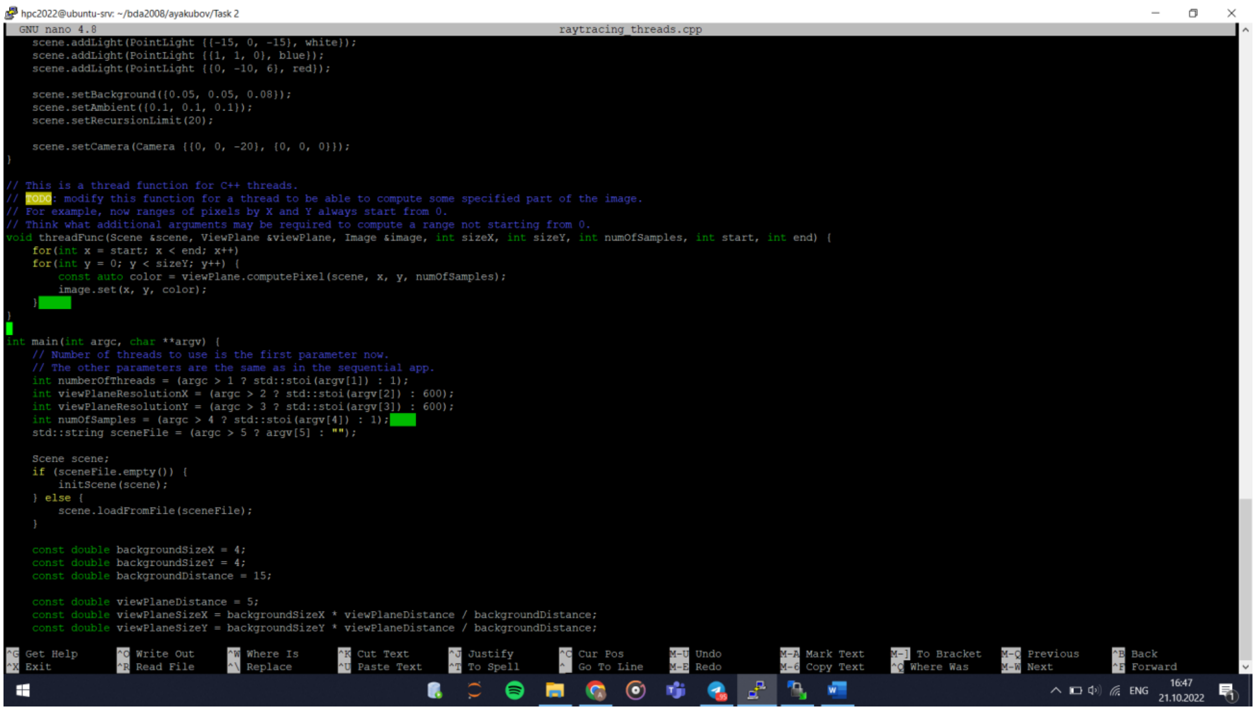


/home/hpc2022/bda2008/ayakubov/Task 2/

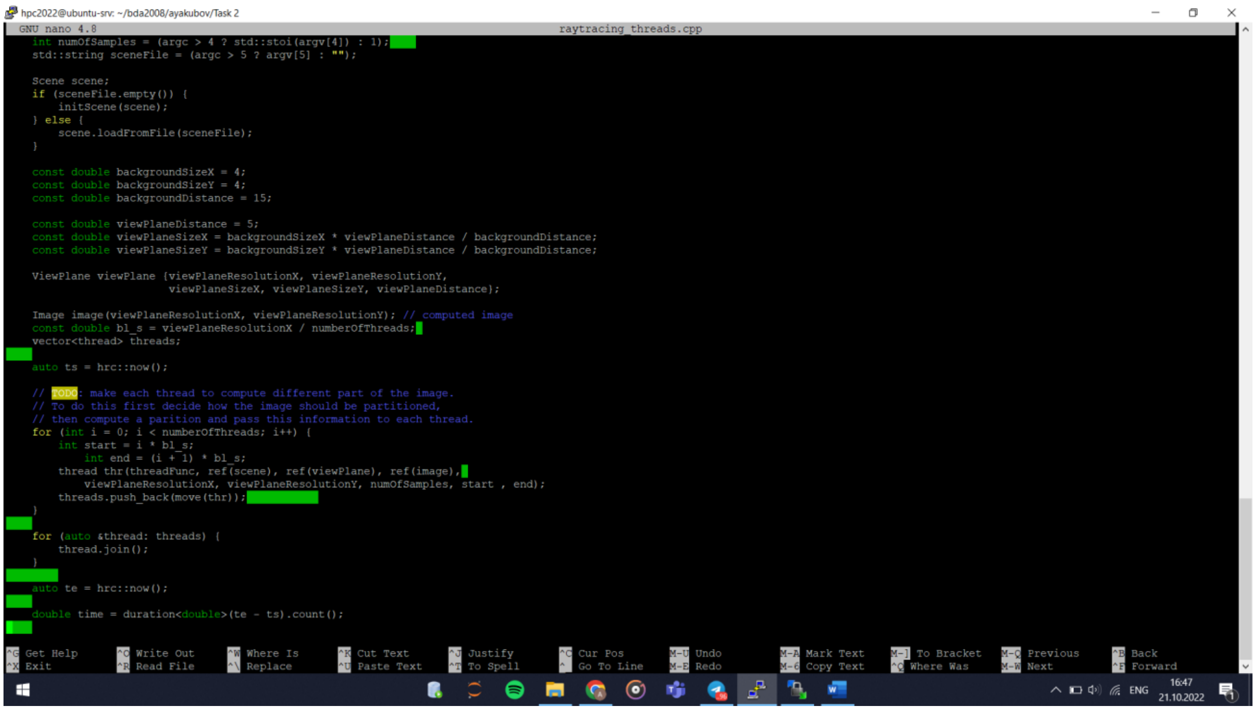
Имя	Размер	Изменено	Права	Владелец
..		20.10.2022 12:52:44	rwxrwxr-x	hpc2022
raytracing_threads.out	208 KB	21.10.2022 16:44:36	rwxrwxr-x	hpc2022
raytracing_threads.cpp	5 KB	21.10.2022 16:44:22	rw-rw-r--	hpc2022
raytracing_16.jpg	29 KB	21.10.2022 16:46:01	rw-rw-r--	hpc2022
raytracing_8.jpg	29 KB	21.10.2022 16:45:52	rw-rw-r--	hpc2022
raytracing_4.jpg	29 KB	21.10.2022 16:45:43	rw-rw-r--	hpc2022
raytracing_2.jpg	29 KB	21.10.2022 16:45:31	rw-rw-r--	hpc2022
raytracing_1.jpg	29 KB	21.10.2022 16:45:18	rw-rw-r--	hpc2022

(Forgot to make it in the beginning)

Step 2.



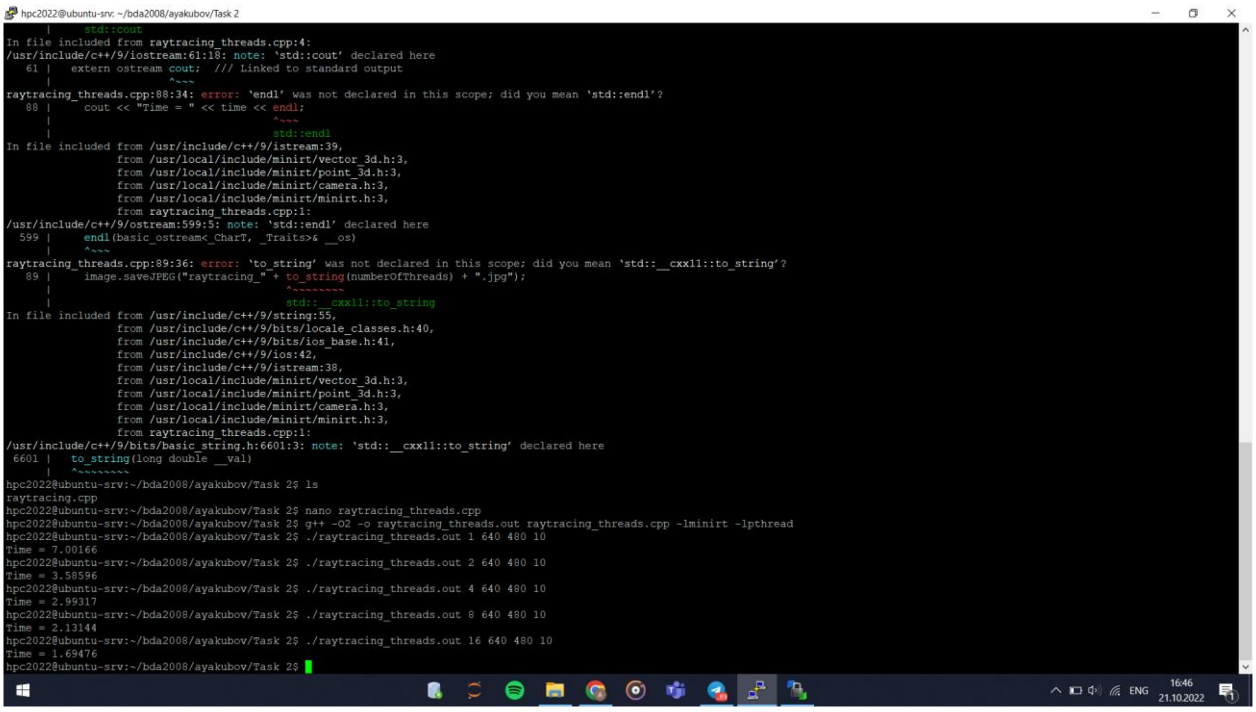
```
  GNU nano 4.8                                             raytracing_threads.cpp
    scene.addLight(PointLight {{-15, 0, -15}, white});
    scene.addLight(PointLight {{1, 1, 0}, blue});
    scene.addLight(PointLight {{0, -10, 6}, red});

    scene.setBackground({0.05, 0.05, 0.08});
    scene.setAmbient({0.1, 0.1, 0.1});
    scene.setRecursionLimit(20);

    scene.setCamera(Camera {{0, 0, -20}, {0, 0, 0}});
}

// This is a thread function for C++ threads.
// TODO: modify this function for a thread to be able to compute some specified part of the image.
// For example, now ranges of pixels by X and Y always start from 0.
// Think what additional arguments may be required to compute a range not starting from 0.
void threadFunc(Scene &scene, ViewPlane &viewPlane, Image &image, int sizeX, int sizeY, int numOfSamples, int start, int end) {
    for(int x = start; x < end; x++)
    for(int y = 0; y < sizeY; y++) {
        const auto color = viewPlane.computePixel(scene, x, y, numOfSamples);
        image.set(x, y, color);
    }
}

int main(int argc, char **argv) {
    // Number of threads to use is the first parameter now.
    // The other parameters are the same as in the sequential app.
    int numberOfThreads = (argc > 1 ? std::stoi(argv[1]) : 1);
    int viewPlaneResolutionX = (argc > 2 ? std::stoi(argv[2]) : 600);
    int viewPlaneResolutionY = (argc > 3 ? std::stoi(argv[3]) : 600);
    int numOfSamples = (argc > 4 ? std::stoi(argv[4]) : 1);
    std::string sceneFile = (argc > 5 ? argv[5] : "");

    Scene scene;
    if (sceneFile.empty()) {
        initScene(scene);
    } else {
        scene.loadFromFile(sceneFile);
    }

    const double backgroundSizeX = 4;
    const double backgroundSizeY = 4;
    const double backgroundDistance = 15;

    const double viewPlaneDistance = 5;
    const double viewPlaneSizeX = backgroundSizeX * viewPlaneDistance / backgroundDistance;
    const double viewPlaneSizeY = backgroundSizeY * viewPlaneDistance / backgroundDistance;

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text  M-] To Bracket M-Q Previous  ^B Back
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text  ^Q Where Was   M-W Next      ^F Forward
```



```
  GNU nano 4.8                                             raytracing_threads.cpp
    int numOfSamples = (argc > 4 ? std::stoi(argv[4]) : 1);
    std::string sceneFile = (argc > 5 ? argv[5] : "");

    Scene scene;
    if (sceneFile.empty()) {
        initScene(scene);
    } else {
        scene.loadFromFile(sceneFile);
    }

    const double backgroundSizeX = 4;
    const double backgroundSizeY = 4;
    const double backgroundDistance = 15;

    const double viewPlaneDistance = 5;
    const double viewPlaneSizeX = backgroundSizeX * viewPlaneDistance / backgroundDistance;
    const double viewPlaneSizeY = backgroundSizeY * viewPlaneDistance / backgroundDistance;

    ViewPlane viewPlane {viewPlaneResolutionX, viewPlaneResolutionY,
                         viewPlaneSizeX, viewPlaneSizeY, viewPlaneDistance};

    Image image(viewPlaneResolutionX, viewPlaneResolutionY); // computed image
    const double bl_s = viewPlaneResolutionX / numberOfThreads;
    vector<thread> threads;

    auto ts = hrc::now();

    // TODO: make each thread to compute different part of the image.
    // To do this first decide how the image should be partitioned,
    // then compute a parition and pass this information to each thread.
    for (int i = 0; i < numberOfThreads; i++) {
        int start = i * bl_s;
            int end = (i + 1) * bl_s;
        thread thr(threadFunc, ref(scene), ref(viewPlane), ref(image),
            viewPlaneResolutionX, viewPlaneResolutionY, numOfSamples, start , end);
        threads.push_back(move(thr));
    }

    for (auto &thread: threads) {
        thread.join();
    }

    auto te = hrc::now();

    double time = duration<double>(te - ts).count();

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text  M-] To Bracket M-Q Previous  ^B Back
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text  ^Q Where Was   M-W Next      ^F Forward
```

Step 3.

a)



```
hpc2022@ubuntu-srv: ~/bda2008/ayakubov/Task 2
      |     std::cout
In file included from raytracing_threads.cpp:4:
/usr/include/c++/9/iostream:61:18: note: 'std::cout' declared here
   61 |   extern ostream cout;  /// Linked to standard output
      |                  ^~~~
raytracing_threads.cpp:88:34: error: 'endl' was not declared in this scope; did you mean 'std::endl'?
   88 |     cout << "Time = " << time << endl;
      |                                  ^~~~
      |                                  std::endl
In file included from /usr/include/c++/9/istream:39,
                 from /usr/local/include/minirt/vector_3d.h:3,
                 from /usr/local/include/minirt/point_3d.h:3,
                 from /usr/local/include/minirt/camera.h:3,
                 from /usr/local/include/minirt/minirt.h:3,
                 from raytracing_threads.cpp:1:
/usr/include/c++/9/ostream:599:5: note: 'std::endl' declared here
  599 |     endl(basic_ostream<_CharT, _Traits>& __os)
      |     ^~~~
raytracing_threads.cpp:89:36: error: 'to_string' was not declared in this scope; did you mean 'std::__cxx11::to_string'?
   89 |     image.saveJPEG("raytracing_" + to_string(numberOfThreads) + ".jpg");
      |                                    ^~~~~~~~~
      |                                    std::__cxx11::to_string
In file included from /usr/include/c++/9/string:55,
                 from /usr/include/c++/9/bits/locale_classes.h:40,
                 from /usr/include/c++/9/bits/ios_base.h:41,
                 from /usr/include/c++/9/ios:42,
                 from /usr/include/c++/9/istream:38,
                 from /usr/local/include/minirt/vector_3d.h:3,
                 from /usr/local/include/minirt/point_3d.h:3,
                 from /usr/local/include/minirt/camera.h:3,
                 from /usr/local/include/minirt/minirt.h:3,
                 from raytracing_threads.cpp:1:
/usr/include/c++/9/bits/basic_string.h:6601:3: note: 'std::__cxx11::to_string' declared here
 6601 |   to_string(long double __val)
      |   ^~~~~~~~~
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ls
raytracing.cpp
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ nano raytracing_threads.cpp
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ g++ -O2 -o raytracing_threads.out raytracing_threads.cpp -lminirt -lpthread
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ./raytracing_threads.out 1 640 480 10
Time = 7.00166
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ./raytracing_threads.out 2 640 480 10
Time = 3.58596
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ./raytracing_threads.out 4 640 480 10
Time = 2.99317
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ./raytracing_threads.out 8 640 480 10
Time = 2.13144
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$ ./raytracing_threads.out 16 640 480 10
Time = 1.69476
hpc2022@ubuntu-srv:~/bda2008/ayakubov/Task 2$
```

b, c)

Number of threads	Execution time	Speedup(N)	Efficiency
1	7.00166	1	1
2	3.58596	1,95252	0,97626
4	2.99317	2,33921	0,58480
8	2.13144	3,28494	0,41061
16	1.69476	4,13135	0,25820

Conclusion.

First, I created bl_s(block_size), start and end, this loop is needed in order to go through the threads. After compiling with arguments, I conclude that the more treads, the less time it takes to compile, regarding speedup(N), on the contrary, it increases, and the efficiency decreases, but efficiency ≤ 1.

Link to GitHub: https://github.com/Am1rrr/hpc/tree/main/Task%202