Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра «Информатика»

**Лабораторная работа № 7**

по дисциплине: **«Операционные Системы и Среды»**

**Синхронизация процессов**

Выполнил студент

группы ИП-31

*Васюков И.А.*

Проверил старший преподаватель

*Самовендюк Н.В.*

Гомель 2019 г.

**Цель:** изучить типовые механизмы синхронизации процессов.

**Практическая часть**

1. Разработать многопоточное приложение с использованием минимум двух потоков и различных средств синхронизации.

|  |  |  |
| --- | --- | --- |
| 4. | Два потока увеличивают значение общей переменной | мьютексы |

2. Время входа в критическую секцию для каждого потока генерировать случайным образом.

3. В процессе работы приложение в консоль должна выводится информация о состоянии потока (работа в некритической секции, работа в критической секции, время входа и выхода из критической секции).

4. Убедиться в результативности применения средств синхронизации потоков, сравнив результаты работы программ с использованием и без использования средств синхронизации.

**Код программы:**

#include <iostream>

#include <mutex>

#include <thread>

#include <chrono>

#include <fstream>

#include <string>

using namespace std;

void addValueOne();

void addValueTwo();

mutex coolMutex;

int x = 0;

int main() {

thread t1(addValueOne);

thread t2(addValueTwo);

t1.join();

t2.join();

return 0;

}

void addValueOne() {

int i = 0;

while (i < 7)

{

coolMutex.lock();

clock\_t start, end;

start = clock();

cout << "Function\_1 start working: " << start << " ms" << endl;

x += 1;

cout << "x = " << x << endl;

coolMutex.unlock();

end = clock();

cout << "Function\_1 finish working: " << end << " ms\nTime work is " << ((double)end - start) << " ms" << endl << endl;

this\_thread::sleep\_for(std::chrono::seconds(rand() % 10));

i++;

}

}

void addValueTwo() {

int i = 0;

while (i < 13)

{

coolMutex.lock();

clock\_t start, end;

start = clock();

cout << "Function\_2 start working: " << start << " ms" << endl;

x += 2;

cout << "x = " << x << endl;

coolMutex.unlock();

end = clock();

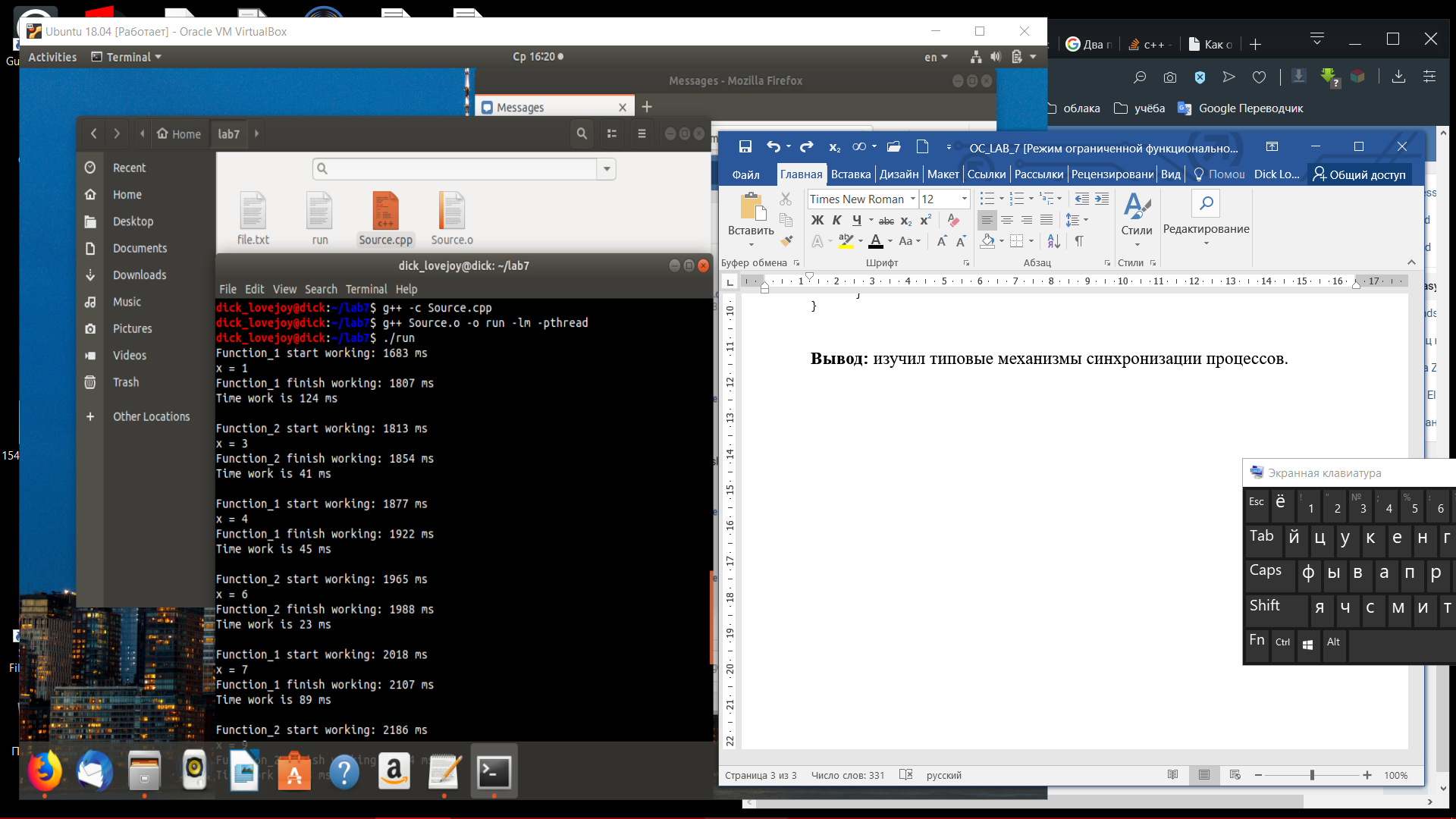
cout << "Function\_2 finish working: " << end << " ms\nTime work is " << ((double)end - start) << " ms" << endl << endl;

this\_thread::sleep\_for(std::chrono::seconds(rand() % 10));

i++;

}

}



**Вывод:** В данной работе были изучены типовые механизмы синхронизации процессов.