МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №4

По предмету «Операционные системы и системное ПО»

«ОРГАНИЗАЦИЯ МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ В ОПЕРАЦИОННОЙ СИСТЕМЕ LINUX»

Выполнил:

студент группы ПО(аб) –71

Маслеников М.В.

Проверил :

Тормозов В.С.

Хабаровск – 2020г.

Цель работы: изучение принципов межпроцессных коммуникаций, организация и проведение сравнительного анализа каналов передачи данных между процессами и потоками в операционной системе Linux.

**Порядок выполнения работы.**

1. Написать программу для работы с неименованными каналами.

Для измерения скорости передачи данных данные меняются от 1 б до 100 мб.

Скорость измеряется следующим образом: объем данных (б) / время (мс).

Код программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define ITER 9 // степень передаваемых данных

int main(int argc, char \*argv[])

{

int fd[2], pid, j;

int nread; //количество записанных и считанных данных в байтах

int sizeB = 1; //размер буфера

char \*buf;

clock\_t start, stop;

double rest; //время в милисекунды

int shmid; // Сегмент общей памяти

key\_t key\_s = ftok(".", 's');

char \*shmPtr;

if((shmid = shmget(key\_s, sizeof(int), 0666|IPC\_CREAT)) == -1)

exit(1);

printf("Сегмент = %d\n", shmid);

shmPtr = shmat(shmid, 0, 0);

if(shmPtr == (char \*) -1)

exit(2);

for(short int i = 0; i < ITER; i++)

{

buf = (char\*)malloc(sizeB \* sizeof(char)); //выделение памяти под буфер

for(j = 0; j < sizeB; j++) //инициализация буфера

buf[j] = 'S';

if(pipe(fd) == -1)

{

perror("pipe failed");

exit(1);

}

if((pid = fork()) < 0)

{

perror("fork failed");

exit(2);

}

if(pid == 0) //child

{

char \*tmp; // буфер для использования данных, считанных из канала

tmp = (char\*)malloc(sizeB \* sizeof(char));

if((shmid = shmget(key\_s, sizeof(int), 0)) == -1)

exit(1);

shmPtr = shmat(shmid, 0, 0);

if(shmPtr == (char \*) -1)

exit(2);

close(fd[1]);

start=buf[0];

nread = read(fd[0], buf, sizeB);

for(j = 0; j < sizeB; j++) //использование данных

tmp[j] = buf[j];

stop = clock(); //время окончания чтения

close(fd[0]);

printf("Количество считанных данных = %d\n", nread);

printf("Время, мс = %.2f\n", 1000.0 \* (stop - start) / CLOCKS\_PER\_SEC);

printf("Скорость = %.2f\n------------------------\n", sizeB / (1000.0 \* (stop - start) / CLOCKS\_PER\_SEC));

free(tmp);

shmdt(shmPtr);

exit(0);

}

else //рarent

{

close(fd[0]);

start = clock(); //время начала записи

buf[0] = start;

nread = write(fd[1], buf, sizeB);

printf("Количество записанных данных = %d\n", nread);

close(fd[1]);

wait(NULL);

}

free(buf);

sizeB \*= 10; // Увеличение размера передаваемых данных

}

shmdt(shmPtr);

if(shmctl(shmid, IPC\_RMID, NULL) == -1 ) // удаление сегмента

exit(-1);

}

Результат работы программы:

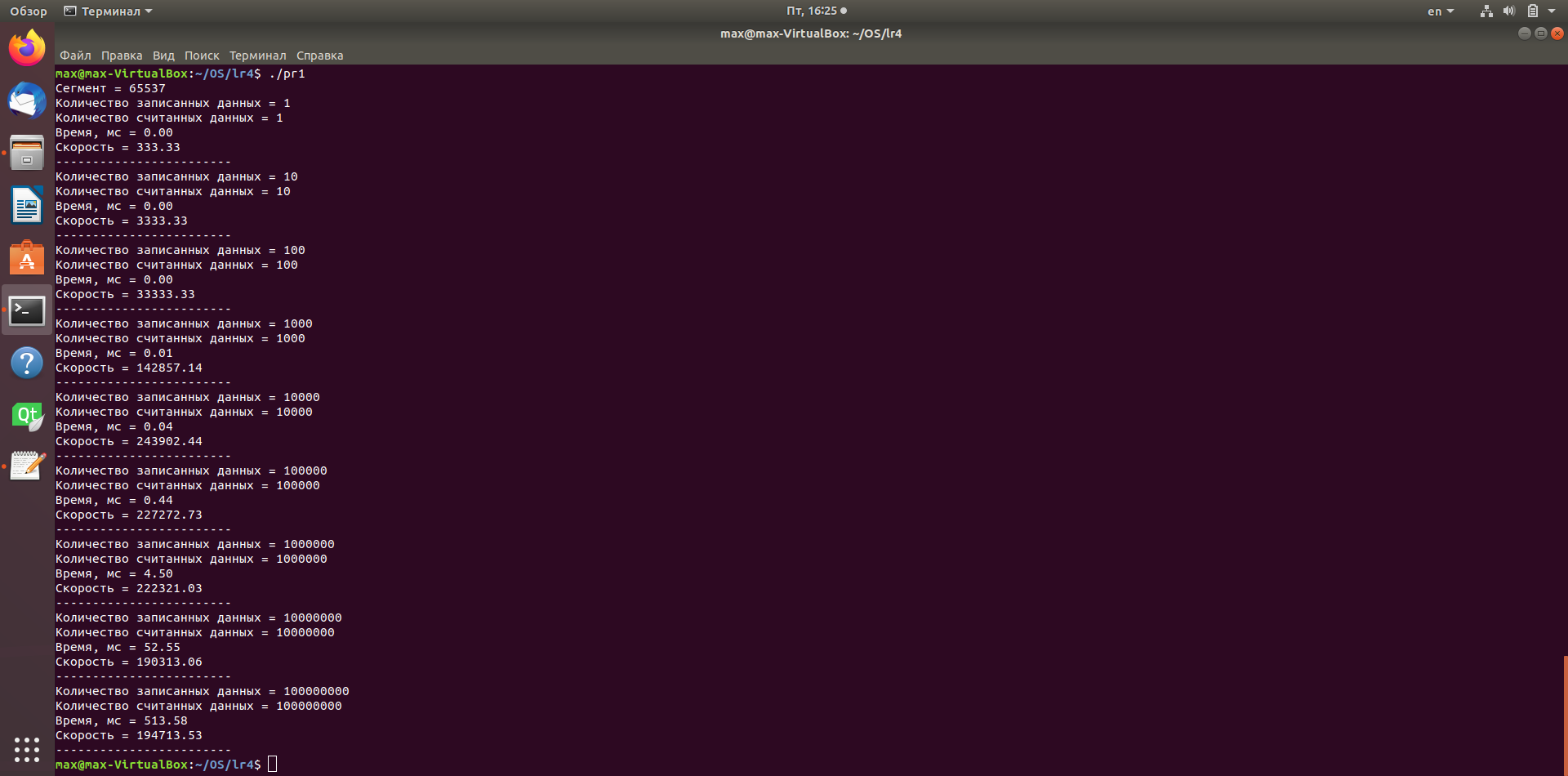
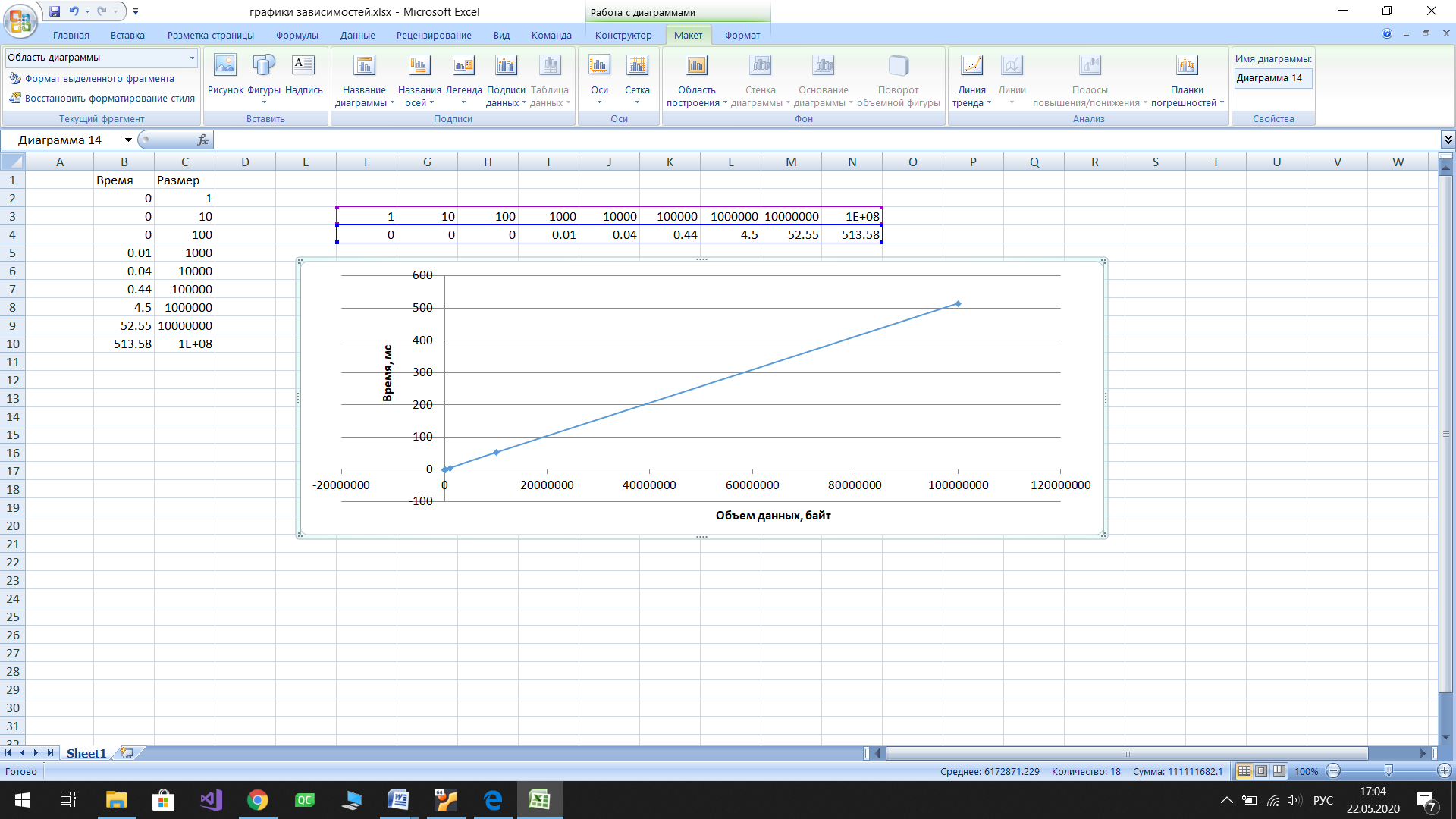


График зависимости времени от объема передаваемых данных:



По графику можно сделать вывод, что зависимость между объемом информации и временем прямопропорциональная. При малом объеме данных (до 1 мб) зависмость трудно отследить, поэтому вывод сделан на основе значений данных, объем которых больше 1мб.

1. Написать программу для работы с общей памятью

Код программы:

#include <stdio.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/wait.h>

#define ITER 9

int main(int argc, char \*argv[])

{

int pid, sizeB = 1, j;

clock\_t t1, t2;

int shmid;

key\_t key\_s = ftok(".", 's');

char \*shmPtr;

double rest;

for(short int i = 0; i < ITER; i++)

{

if((pid = fork()) < 0)

{

perror("fork failed");

exit(2);

}

if(pid == 0)

{

char \*buf;

buf = (char\*)malloc(sizeB \* sizeof(char));

t1 = clock();

shmid = shmget(key\_s, sizeB, 0);

if(shmid == -1)

exit(1);

shmPtr = shmat(shmid, 0, 0);

if (shmPtr == (char \*) -1)

exit(2);

for(j = 0; j< sizeB; j++)

buf[j] = shmPtr[j];

t2 = clock();

rest = 1000.0 \* (t2 - t1) / CLOCKS\_PER\_SEC + shmPtr[0];

printf("Сегмент %d = %d, размер = %d\n", i, shmid, sizeB);

printf("Время, мс = %.2f\n", rest);

printf("Скорость = %.2f\n------------------------\n", sizeB / rest);

exit(0);

}

else

{

t1 = clock();

if((shmid = shmget(key\_s, sizeB, 0666|IPC\_CREAT)) == -1)

exit(1);

printf("Сегмент %d = %d, размер = %d\n", i, shmid, sizeB);

shmPtr = shmat(shmid, 0, 0);

if(shmPtr == (char \*) -1)

exit(2);

for(j = 0; j < sizeB; j++)

shmPtr[j] = 'S';

t2 = clock();

rest = 1000.0 \* (t2 - t1) / CLOCKS\_PER\_SEC;

shmPtr[0] = rest;

wait(NULL);

shmdt(shmPtr);

if(shmctl(shmid, IPC\_RMID, NULL) == -1 ) // удаление сегмента

exit(-1);

}

sizeB \*= 10;

}

}

Результат работы программы:

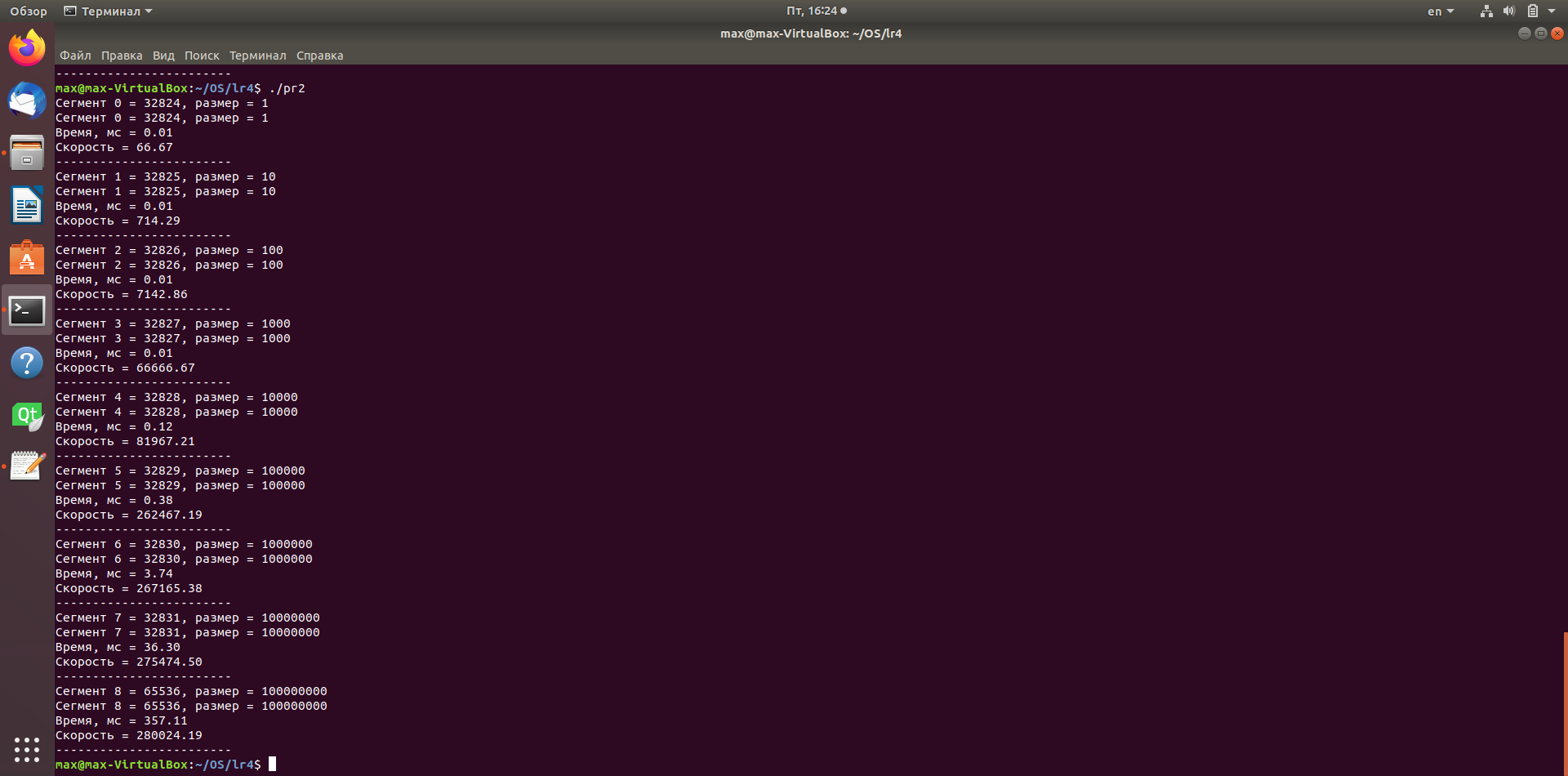
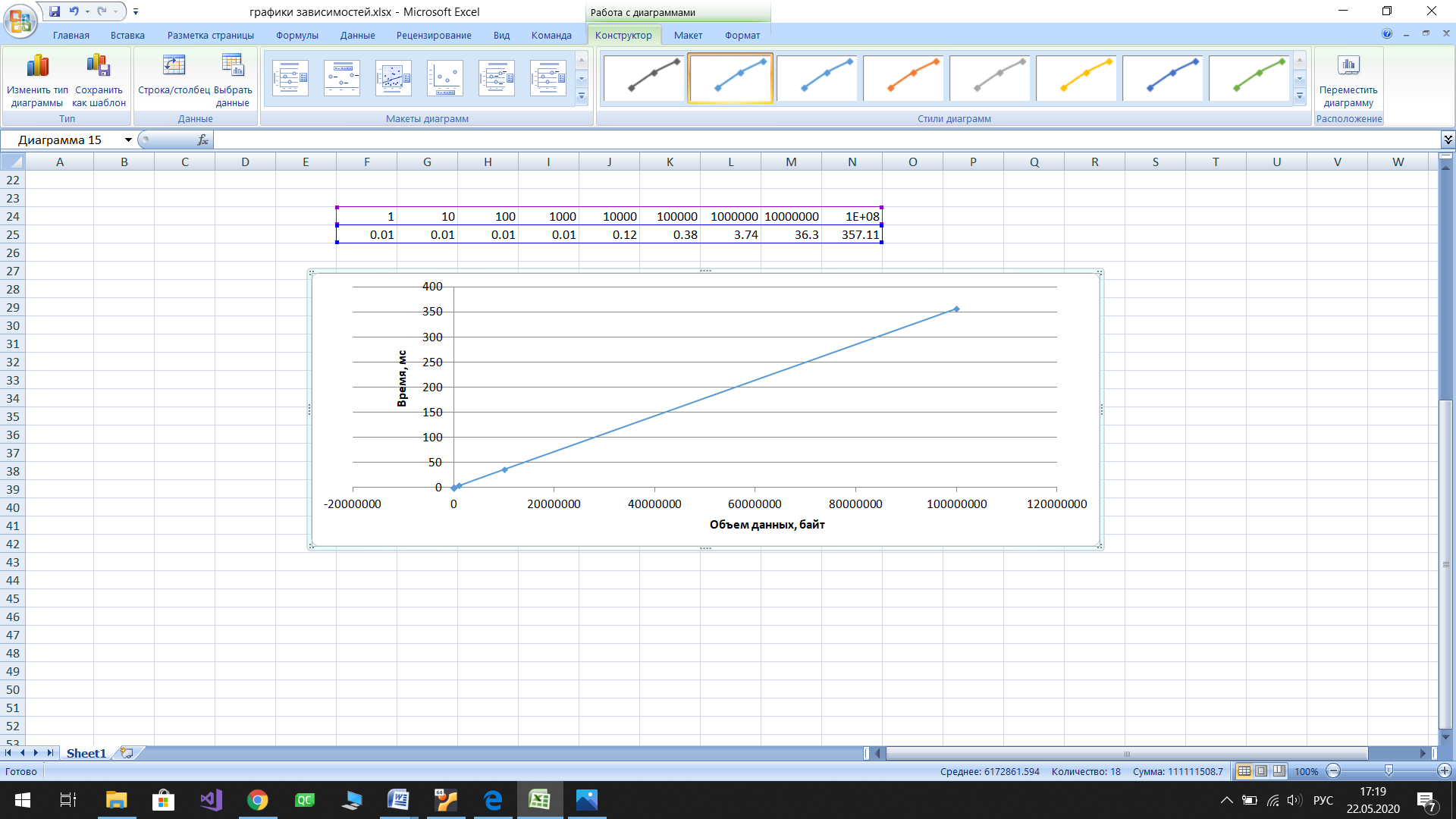


График зависимости времени от объема передаваемых данных:



По графику можно сделать вывод, что он также, как и график неименованного канала, имеет прямую зависимость, при этом время на передачу данных одинакового объема при помощи общего сегмента памяти меньше, чем время на передачу при помощи неименованного канала. Это означает, что скорость передачи при использовании общего сегмента памяти больше, чем при использовании неименованного канала.