Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №2

«Процессы и асинхронное взаимодействие. Часть 3, 4, 5»

Выполнил: студент 4 курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: старший преподаватель кафедры ПМиК

Милешко А.В.

Новосибирск, 2020 г.

Задание

3. **Заставить нарисованные элементы двигаться независимо друг от друга с помощью параллельных процессов (можно изменять во времени положение, цвет, размеры, конфигурацию графических элементов). Предусмотреть завершение программы по нажатию на любую клавишу.**

Выполнение задания

Для выполнения данного задания за основу был взял 1 пункт лабораторной. Для выполнения движения была использована функция MoveTo(x, y, figure) – принимающая в себя координаты x и y и фигуру, которую необходимо двигать. Далее для реализации параллельных процессов был использован метод fork() из библиотеки unistd.h. Было реализовано вложенное создание процессов-родителей для реализации 4 разных процессов чтобы определить в каждый по 2 фигуры.

Так же с помощью InputChar() – было реализовано закрытие программы по нажатию любой из клавиш.

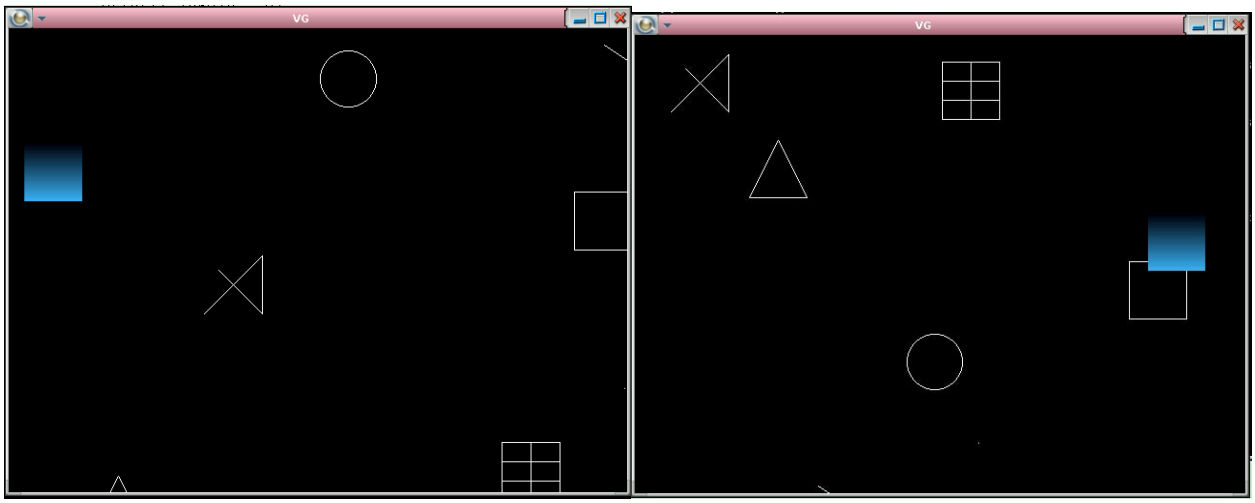

Рис.1 – демонстрация разных этапов работы программы

Финальный код программы находится в приложении.

4. **Нарисовать нечто, движущееся по замкнутой кривой. Организовать изменение траектории движения по нажатию на клавиши (организуя взаимодействие процессов через общую область памяти (shared memory)). В качестве фона можно использовать (оживленную) картину, созданную на предыдущих этапах работы.**

Для выполнения данного задания была реализована программа с использованием shared-memory, для инициализации такой памяти был

использован метод mmap с определенными флагами – MAP_SHARED, MAP_ANONYMOUS, влияющие на тип собственно памяти – разделяемая и анонимная.

Далее в программе было реализовано движение фигуры квадрат по Улитке Паскаля, теория которой была описана в задании к лабораторной. С помощью стрелок производится смещение «Улитки» в соответствующую сторону, что продемонстрированно на рисунке 2.

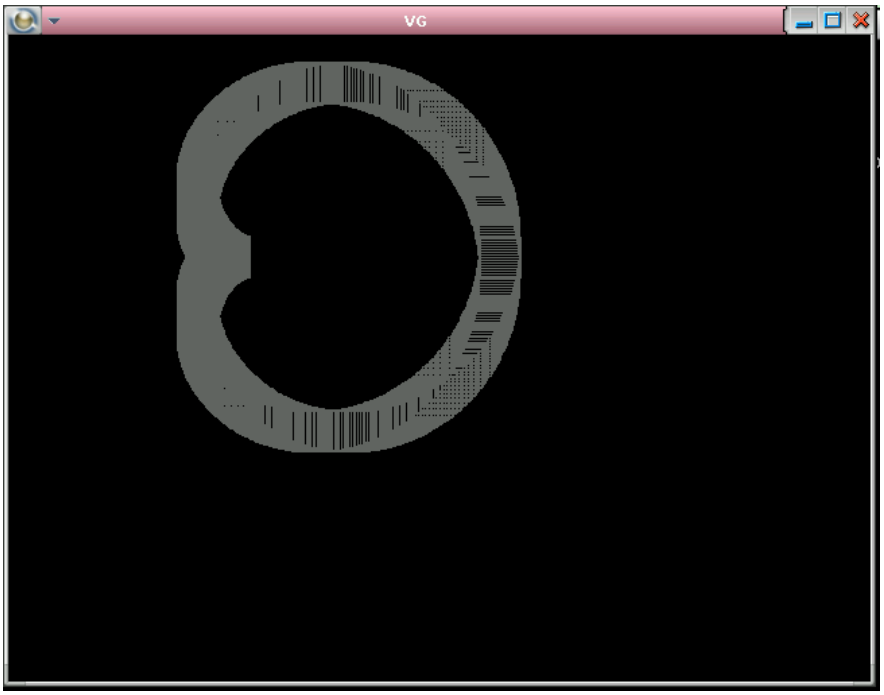

Рис.1

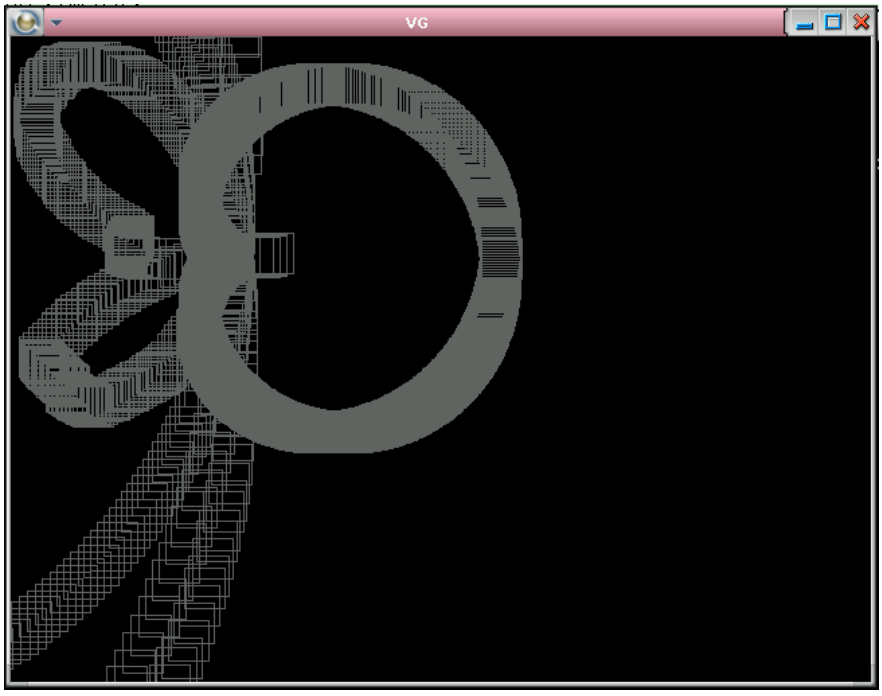

Рис.2

Листинг программы в приложении.

5. **Затем последнюю программу сделать с помощью нитей в одном процессе.**

В программу были успешно внедрены нити из библиотеки <pthread.h>. Циклы отрисовки и движения были разнесены в разные функции для нитей. Далее в main() были инициализированы две нити – mover и drawer соответственно. Потом была запущена нить mover, которую прерывает нить drawer при ее использовании.

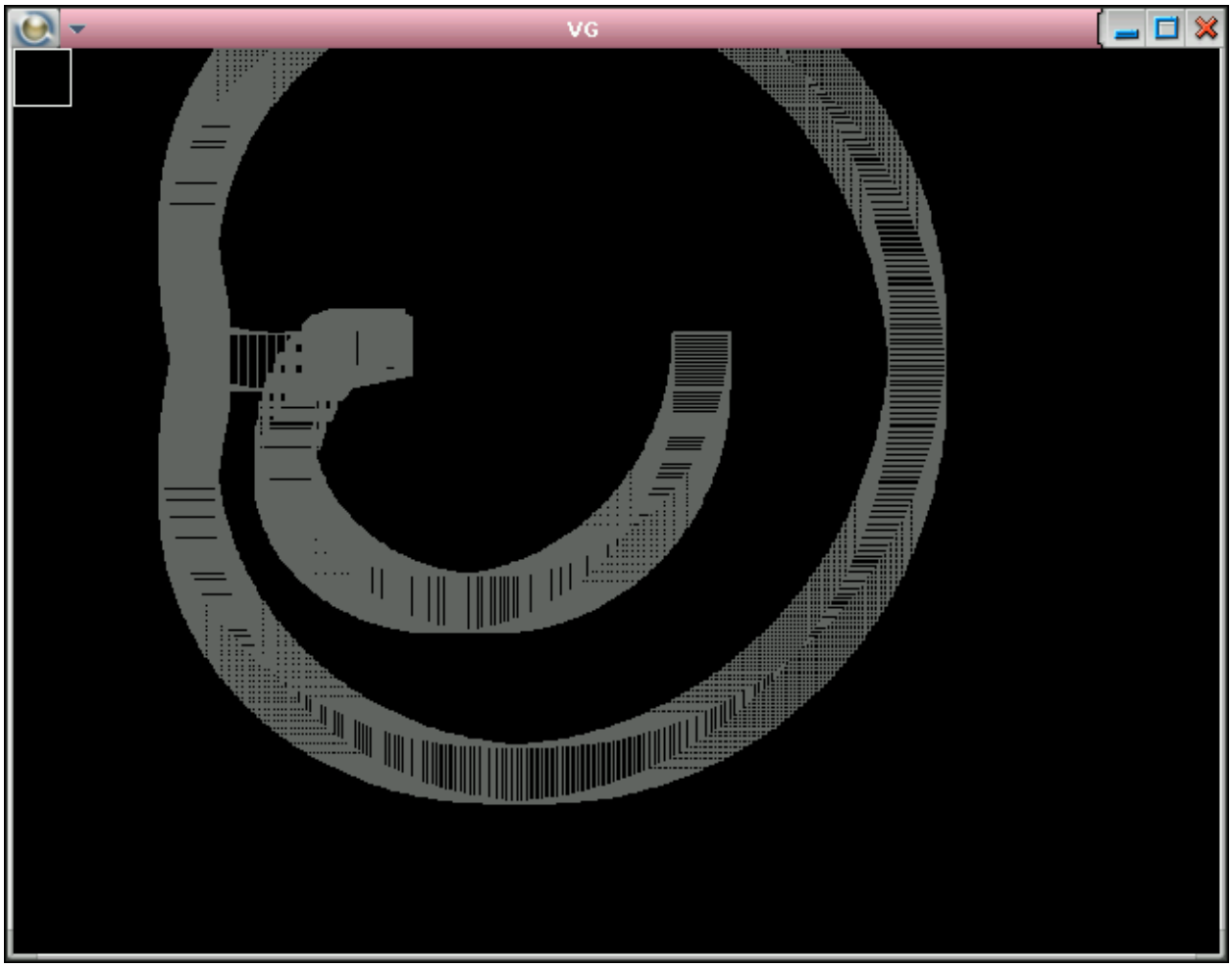

Рис.1 – демонстрация работы программы со сдвинутой улиткой паскаля.

Листинг находится в приложении.

ПРИЛОЖЕНИЕ. ЛИСТИНГ ПРОГРАММ:

3.cpp:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <vingraph.h>
```

```
#include <time.h>

int main()
{
        srand(time(0));
        ConnectGraph();

        int pixel = Pixel(20, 40);

        int line = Line (70, 50, 130, 90);

        tPoint p1 [] = { {140, 100}, {200, 40}, {200, 100}, {155, 55} };
        int polyline = Polyline (p1, 4);

        int rect = Rect (210, 40, 60, 60);

        tPoint p2 [] = { {280, 100}, {310, 40}, {340, 100} };
        int polygon = Polygon (p2, 3);

        int ellipse = Ellipse (350, 40, 60, 60);

        int grid = Grid (560, 40, 60, 60, 3, 2);

        int *im_buf = (int*) malloc (60*60*4);
        for (int i = 0, c = 10; i < 60; i++, c += 0x010304)
                for (int j = 0; j < 60; j++)
                        im_buf [60*i + j] = c;
        int im32 = Image32 (630, 40, 60, 60, im_buf);

        tPoint dim;
        dim = GetDim(0);
        delay(1000);

        int a = getpid();
        printf("\n%d\n", a);

        pid_t proc1, proc2, proc3, proc4;

        if(proc1 = fork())
        {
                int b = getpid();
                printf("\n%d\n", b);
                if(proc2 = fork())
                {
                        int c = getpid();
                        printf("\n%d\n", c);
                        if(proc3 = fork())
                        {
                                int d = getpid();
                                printf("\n%d\n", d);
                                if(proc4 = fork())
                                {
                                        InputChar();
                                        int e = getpid();
                                        printf("\n%d\n", e);
                                        CloseGraph();
                                }
                                else
                                {
                                        srand(time(0));
                                        while(true)
                                        {
                                                int way_x = rand()%dim.x, way_y
= rand()%dim.y;
```

```
                                                MoveTo(way_x, way_y, pixel);
                                                way_x = rand()%dim.x, way_y =
rand()%dim.y;
                                                MoveTo(way_x, way_y, line);
                                                delay(400);
                                        }
                                }
                        }
                        else
                        {
                                srand(time(0));
                                while(true)
                                {
                                        int way_x = rand()%dim.x, way_y =
rand()%dim.y;
                                        MoveTo(way_x, way_y, polyline);
                                        way_x = rand()%dim.x, way_y =
rand()%dim.y;
                                        MoveTo(way_x, way_y, rect);
                                        delay(600);
                                }
                        }
                }
                else
                {
                        srand(time(0));
                        while(true)
                        {
                                int way_x = rand()%dim.x, way_y = rand()%dim.y;
                                MoveTo(way_x, way_y, polygon);
                                way_x = rand()%dim.x, way_y = rand()%dim.y;
                                MoveTo(way_x, way_y, ellipse);
                                delay(800);
                        }
                }
        }
        else
        {
                srand(time(0));
                while(true)
                {
                        int way_x = rand()%dim.x, way_y = rand()%dim.y;
                        MoveTo(way_x, way_y, grid);
                        way_x = rand()%dim.x, way_y = rand()%dim.y;
                        MoveTo(way_x, way_y, im32);
                        delay(1000);
                }
        }
        CloseGraph();
        return 0;
}
```

4.cpp:

```
#include <cmath>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <unistd.h>
#include <vingraph.h>
#include <time.h>

using namespace std;
```

```
int main()
{
        ConnectGraph();
        int key;
        float* gvara = static_cast<float*>(mmap(NULL, sizeof* gvara,
PROT_READ|PROT_WRITE,

            MAP_SHARED|MAP_ANONYMOUS,

            -1,0));
        float* gvarb = static_cast<float*>(mmap(NULL, sizeof* gvarb,
PROT_READ|PROT_WRITE,

            MAP_SHARED|MAP_ANONYMOUS,

            -1,0));
        *gvara = 100;
        *gvarb = 100;
        int x, y;
        float rho, a, phi = 0,b;

        pid_t proc;
        if(proc = fork())
        {
                while(true)
                {
                        key = InputChar();
                        if(key == ' ')
                                break;
                        if(key == 81)
                                *gvarb = *gvarb + 5;
                        if(key == 82)
                                *gvara = *gvara + 5;
                        if(key == 83)
                                *gvarb = *gvarb - 5;
                        if(key == 84)
                                *gvara = *gvara - 5;
                }
                CloseGraph();
                return 0;
        }
        else
        {
                while(true)
                {
                        Rect (x, y, 30, 30);
                        rho = *gvara * cos(phi) + *gvarb;
                        x = rho * cos(phi) + 150;
                        y = rho * sin(phi) + 150;
                        phi += 0.01;
                        SetColor(RGB(100, 100, 100));
                        Rect (x, y, 30, 30);
                        delay(15);
                }
        }

        return 0;
}
```

5.cpp:

```
#include <cmath>
#include <iostream>
#include <stdlib.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <unistd.h>
#include <vingraph.h>
#include <time.h>

using namespace std;
float* sh_a;
float* sh_b;
int x, y;
float rho, phi = 0;

void *fig_mover(void *args)
{
        static int key;
        while(true)
        {
                key = InputChar();
                if(key == ' ')
                        break;
                if(key == 81)
                        *sh_b = *sh_b + 5;
                if(key == 82)
                        *sh_a = *sh_a + 5;
                if(key == 83)
                        *sh_b = *sh_b - 5;
                if(key == 84)
                        *sh_a = *sh_a - 5;
        }
}

void *fig_drawer(void *args)
{
        while(true)
        {
                Rect (x, y, 30, 30);
                rho = *sh_a * cos(phi) + *sh_b;
                x = rho * cos(phi) + 150;
                y = rho * sin(phi) + 150;
                phi += 0.01;
                SetColor(RGB(100, 100, 100));
                Rect (x, y, 30, 30);
                delay(15);
        }
}

int main()
{
        ConnectGraph();
        sh_a = static_cast<float*>(mmap(NULL, sizeof* sh_a,
PROT_READ|PROT_WRITE,

            MAP_SHARED|MAP_ANONYMOUS,

            -1,0));
        sh_b = static_cast<float*>(mmap(NULL, sizeof* sh_b,
PROT_READ|PROT_WRITE,
```

```
            MAP_SHARED|MAP_ANONYMOUS,

            -1,0));
        *sh_a = 100;
        *sh_b = 100;

        pthread_t mover, drawer;
        pthread_create(&mover, NULL, &fig_mover, NULL);
        pthread_create(&drawer, NULL, &fig_drawer, NULL);
        if(!pthread_join(mover, 0))
                cout << "Code is" << pthread_cancel(drawer);
        CloseGraph();

        return 0;
}
```