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

Лабораторная работа №3

«Взаимодействие между нитями с помощью сообщений»

Выполнил: студент 4 курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: старший преподаватель кафедры ПМиК

Милешко А.В.

Новосибирск, 2020 г.

Задание

1. **Электростанция состоит из следующих элементов: хранилище топлива (1 шт.), транспортное средство (1 шт.), котлы (4 шт.). Элементы станции работают параллельно, каждый по своей программе (что может быть реализовано с помощью нитей). Транспортное средство доставляет топливо из хранилища к котлам. Топливо имеет различные марки (от 1 до 10). Топливо марки 10 горит в котле 10 с (условно), в то время как топливо марки 1 горит всего 1 с. Необходимо написать программу, моделирующую работу электростанции и показывающую на экране процесс ее функционирования.**

Выполнение задания

При выполнении задания были задействованы нити (потоки) из библиотеки <pthread.h> и функции обмена сообщениями между ними. Так как сообщения нельзя передать напрямую были задействованы специальные каналы и соединения. Для создания такого канала используется функция ChannelCreate(unsigned flags), принимающая лишь флаг, потом уже можно вызвать функции MsgRecieve((int chid, void * msg, int bytes, struct _msg_info * info) – принимает chid – id канала который мы получаем после вызова ChannelCreate(), msg – указатель на буфер сообщения, bytes – размер данного буфера, info – указатель на структуру, в которой может храниться доп. информация о сообщении.

Чтобы принимающий поток мог получить сообщение необходимо присоединиться к каналу отдающему через ConnectAttach(uint32_t nd, _t pid, int chid, index, int flags), а потом уже вызывать MsgSend(int coid, const void* smsg, int sbytes, * rmsg, int rbytes) с параметрами: coid – ID, полученный от ConnectAttach. smsg – указатель на буфер сообщения, sbytes – кол-во байтов на отправку, rmsg – указатель на буфер с ответом, rbytes – размер буфера ответа.

В коде работа транспорта, склада и котлов реализовано в разных нитях (потоках), графическое представление работы программы реализовано через библиотеку VinGraph.h.

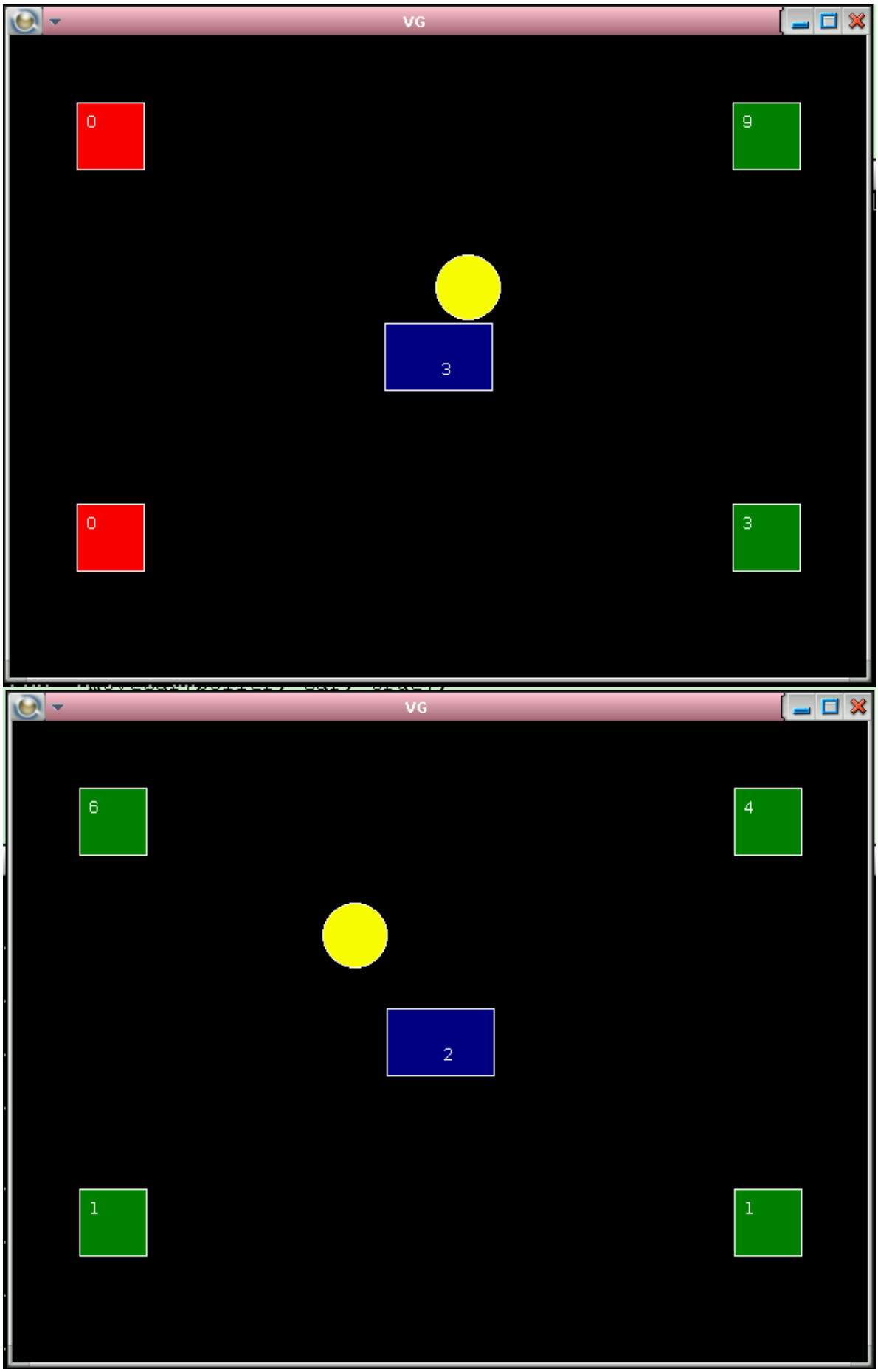


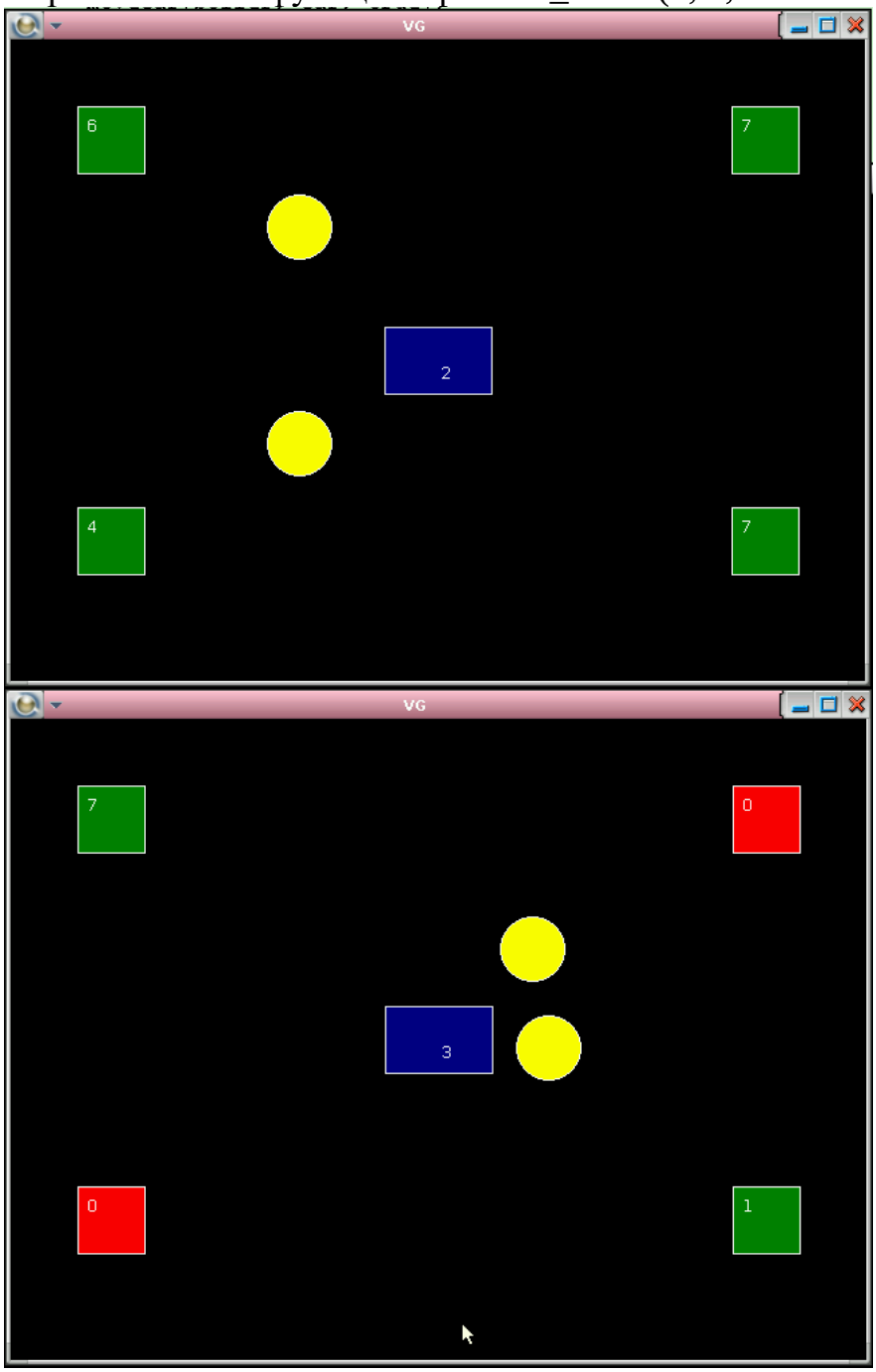
Рис.1 – демонстрация работы программы

Задание

2. **А теперь добавьте второе транспортное средство.**

Выполнение задания

Второе транспортное средство легко добавить через добавление дополнительного потока с автомобилем в функции main() с помощью повторения вызова функции - pthread_create(0, 0, carThread, 0);



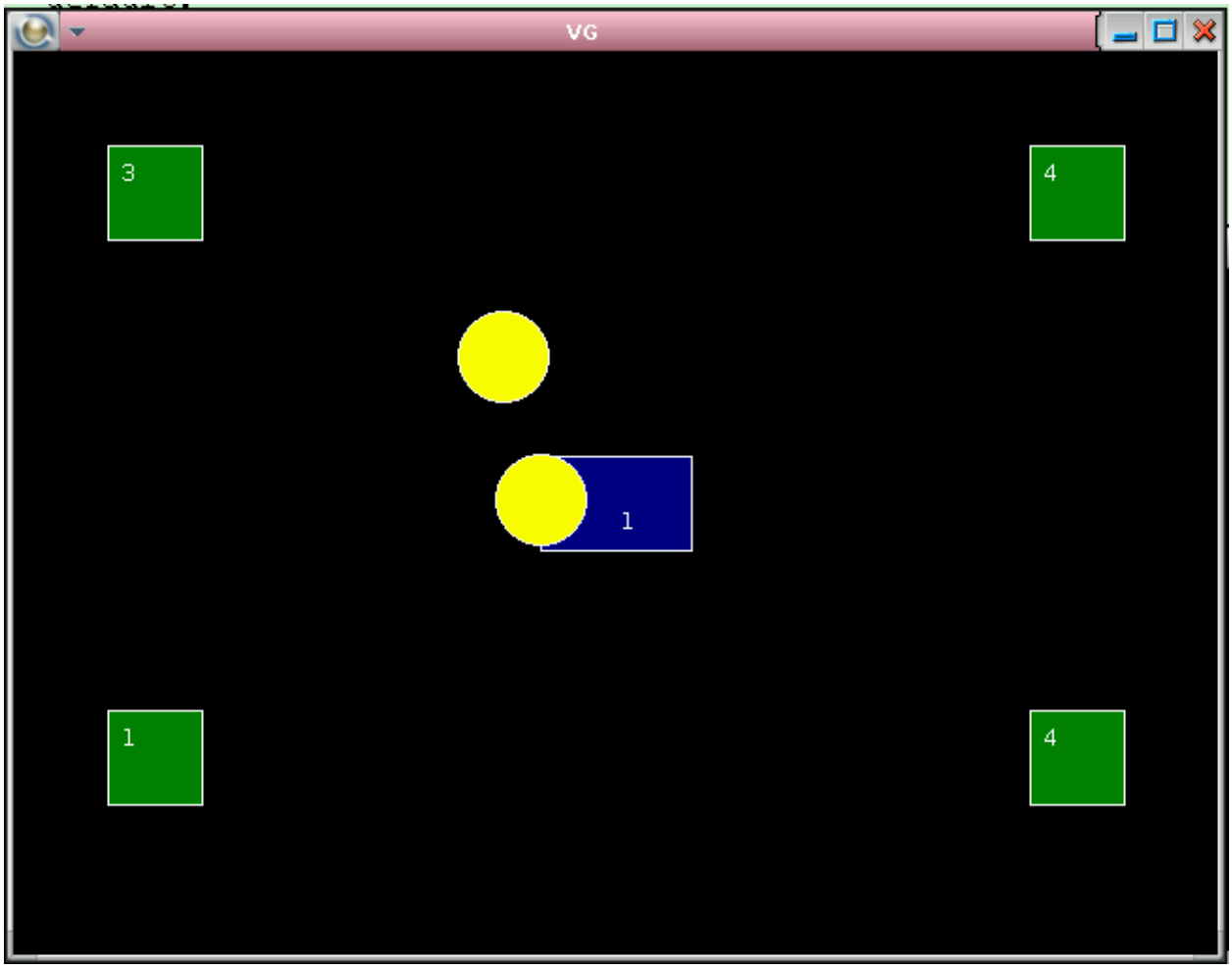
Задание

3. **(использование импульсов) Регулируя скорости работы элементов электростанции, вы можете создать ситуацию, когда котлы будут**

простаивать из-за низкой скорости подвоза топлива. Создайте такую ситуацию. Теперь сделайте так, чтобы топливо подвозилось к котлам заранее, до момента их полной остановки. Это можно реализовать, если котлы будут сообщать о том, что топливо скоро кончится (например, его осталось на 2 с работы). Ясно, что котлы могут это сделать с помощью импульса, т.к. обычное сообщение их заблокировало бы, в то время как они должны продолжать работать.

Выполнение задания

Для реализации данного задания были использованы pulse-методы, которые оповещают потоки о необходимости доставки для котлов в которых топлива осталось меньше чем на 3 секунды, чтобы отправить пульс используется функция MsgSendPulse(int coid, int priority, int code, int value) – отправляет импульс к потоку.



Как видно транспорт уже поехал к бойлеру, в котором топливу осталось гореть 3 секунды.

Листинг готовой программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
```

```
#include <unistd.h>
#include <sys/neutrino.h>
#include <sched.h>
#include <vingraph.h>

#define SPEED_CAR 3
#define POSX_CAR 300
#define POSY_CAR 170

int boiler1 = 0;
int boiler2 = 0;
int boiler3 = 0;
int boiler4 = 0;

int chid1 = ChannelCreate(0);
int chid2 = ChannelCreate(0);

int graphid = ConnectGraph();

void *boilerRoom(void *arg)
{
        int coid = ConnectAttach(0, 0, chid1, 0, 0);
        int numStation = *(int *) arg;
        setprio(0, numStation);
        sched_yield();
        int boiler, mark;
        switch(numStation)
        {
                case 1:
                        boiler = boiler1;
                        break;
                case 2:
                        boiler = boiler2;
                        break;
                case 3:
                        boiler = boiler3;
                        break;
                case 4:
                        boiler = boiler4;
                        break;
                default:
                        printf("Error! Can't define boiler.\n");
                        break;
        }
        tPoint pos = GetPos(boiler);
        int text = Text(pos.x + 5, pos.y + 5, "0\0");
        while(1)
        {
                bool impulse = false;
                Fill(boiler, RGB(255, 0, 0));
                MsgSend(coid, &numStation, sizeof(int), &mark, sizeof(int));
                Fill(boiler, RGB(0, 128, 0));
                char tmp[3] = {};
                while(mark > 0)
                {
                        sprintf(tmp, "%d", mark);
                        SetText(text, tmp);
                        if(mark <= 3 && impulse != true)
                        {
                                MsgSendPulse(coid, numStation, 0, numStation);
                                impulse = true;
                        }
                        usleep(500000);
                        mark--;
```

```
                }
                SetText(text, "0\0");
        }
}

void moveCar(int boiler, int car, bool dir)
{
        int toX = 0, toY = 0;
        tPoint posBoiler = GetPos(boiler);
        tPoint dimBoiler = GetDim(boiler);
        tPoint posCar = GetPos(car);
        if (dir)
        {
                toX = posBoiler.x + dimBoiler.x;
                toY = posBoiler.y + dimBoiler.y / 2;
        }
        else
        {
                toX = POSX_CAR;
                toY = POSY_CAR;
        }
        float dx = (toX - posCar.x) / 50.;
        float dy = (toY - posCar.y) / 50.;
        int tX = abs((int)(dx * 100) % 100);
        int tY = abs((int)(dy * 100) % 100);
        if(tX >= 50)
                if(dx < 0)
                        dx--;
                else
                        dx++;
        if(tY >= 50)
                if(dy < 0)
                        dy--;
                else
                        dy++;
        for(int i = 0; i < 50; i++)
        {
                Move(car, dx, dy);
                usleep(10000);
        }
}

void *fuelTankThread(void *arg)
{
        int coid = ConnectAttach(0, 0, chid2, 0, 0);
        int fuelTank = Rect(280, 215, 80, 50);
        Fill(fuelTank, RGB(0, 0, 128));
        int text = Text(320, 240, "0\0");
        int mark, ready;
        char tmp[3] = {};
        while(1)
        {
                mark = 1 + rand()%10;
                sprintf(tmp, "%d", mark);
                SetText(text, tmp);
                int rcvid = MsgReceive(chid2, 0, sizeof(int), 0);
                MsgReply(rcvid, 0, &mark, sizeof(int));
        }
}

void *carThread(void *arg)
{
        int car = Ellipse(300, 170, 50, 50);
        Fill(car, RGB(255, 255, 0));
```

```
        int point, boiler, rcvid, t;
        int coid1 = ConnectAttach(0, 0, chid1, 0, 0);
        int coid2 = ConnectAttach(0, 0, chid2, 0, 0);
        while(1)
        {
                rcvid = MsgReceive(chid1, &point, sizeof(int), 0);
                boiler = 0;
                switch(point)
                {
                        case 1:
                                boiler = boiler1;
                                break;
                        case 2:
                                boiler = boiler2;
                                break;
                        case 3:
                                boiler = boiler3;
                                break;
                        case 4:
                                boiler = boiler4;
                                break;
                        default:
                                printf("Error! Can't define boiler.\n");
                                break;
                }
                int mark = 0;
                if(boiler != 0)
                {
                        MsgSend(coid2, 0, 0, &mark, sizeof(int));
                        moveCar(boiler, car, true);
                        if(rcvid)
                                MsgReply(rcvid, 0, &mark, sizeof(int));
                        else
                                MsgSendPulse(coid1, mark, 0, mark);
                        moveCar(boiler, car, false);
                }
                else
                        printf("Error! Can't move.\n");
        }
}

int main()
{
        boiler1 = Rect(50, 50, 50, 50);
        boiler2 = Rect(50, 350, 50, 50);
        boiler3 = Rect(540, 50, 50, 50);
        boiler4 = Rect(540, 350, 50, 50);

        int num1 = 1;
        pthread_create(0, 0, boilerRoom, &num1);
        int num2 = 2;
        pthread_create(0, 0, boilerRoom, &num2);
    int num3 = 3;
        pthread_create(0, 0, boilerRoom, &num3);
        int num4 = 4;
        pthread_create(0, 0, boilerRoom, &num4);

        pthread_create(0, 0, fuelTankThread, 0);
        pthread_create(0, 0, carThread, 0);
        pthread_create(0, 0, carThread, 0);
        InputChar();
        CloseGraph();
        return 0;
}
```