ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ЛАБОРАТОРНАЯ РАБОТА № 3

Выполнил:
Студент группы: ИП-712
Алексеев С.В.

Проверил: профессор кафедры ПМиК
Фионов А.Н.

Новосибирск – 2020

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. Используя функции библиотеки VinGraph, нарисовать абстрактную картину, которой представлены (почти) все доступные графические элементы.

Для реализации данного задания необходимо написать программу с обменом сообщениями (между потоками) и использованием POSIX Threads (pthread.h).

Сообщения не передаются между потоками напрямую, поэтому для обмена сообщениями между потоками используются каналы и соединения. Канал можно создать с помощью функции ChannelCreate(). После этого можно вызывать MsgReceive() и MsgReply().

ChannelCreate(unsigned flags) принимает один аргумент-флаг или комбинацию из них.
MsgReceive(int chid, void * msg, int bytes, struct _msg_info * info) принимает 4 аргумента. chid, ID, который мы получаем после вызова ChannelCreate(), msg – указатель на буфер с данными, bytes – размер буфера, info – NULL или указатель на структуру _msg_info, где может хранится дополнительная информация о сообщении.

Принимающий поток может создать соединение к каналу другого потока используя ConnectAttach(), а потом уже вызывать MsgSend(). Получается своеобразная схема “клиент-сервер’’.

ConnectAttach(uint32_t nd, _t pid, int chid, index, int flags) устанавливает соединение между потоком и каналом. nd – дескриптор, pid – ID процесса-владельца каналом, chid - chid, ID, который мы получаем после вызова ChannelCreate(), index – наименее возможный ID соединения, flags – флаги.
MsgSend(int coid, const void* smsg, int sbytes, * rmsg, int rbytes) отправка сообщения в канал. 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.

Деятельность транспорта (carThread), склада топлива (fuelTankThread) и котлов (boilerRoom) реализована в разных потоках. Графика реализована с помощью VinGraph и таймера для отображения анимации.

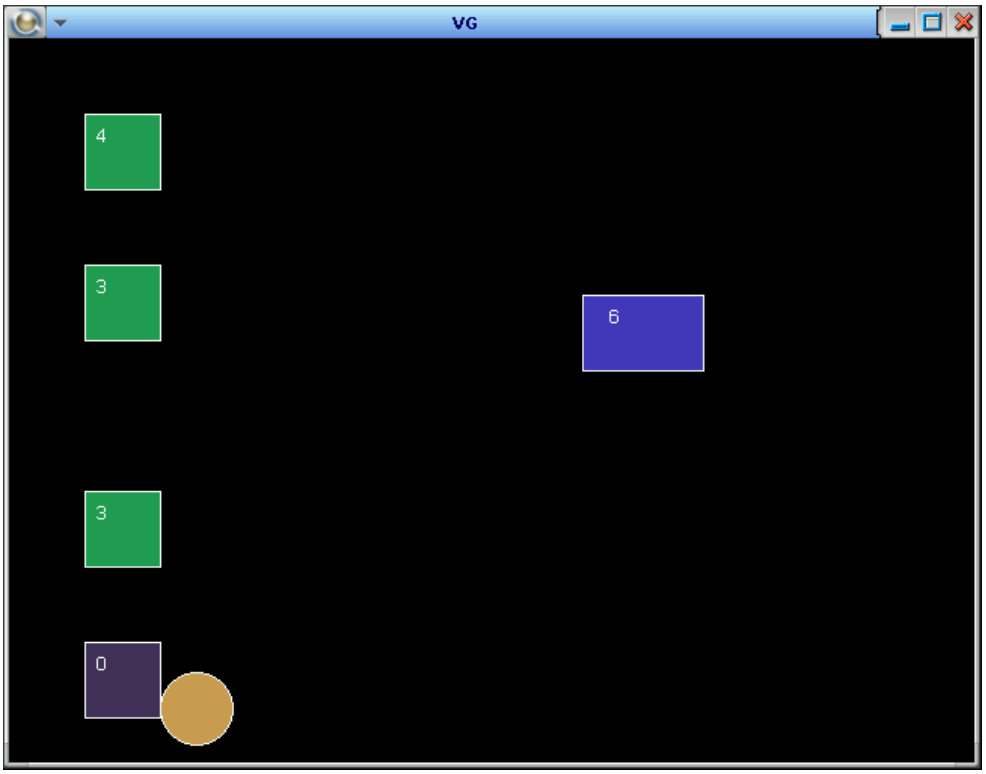

Рис.1 – Процесс работы программы с одним транспортным потоком

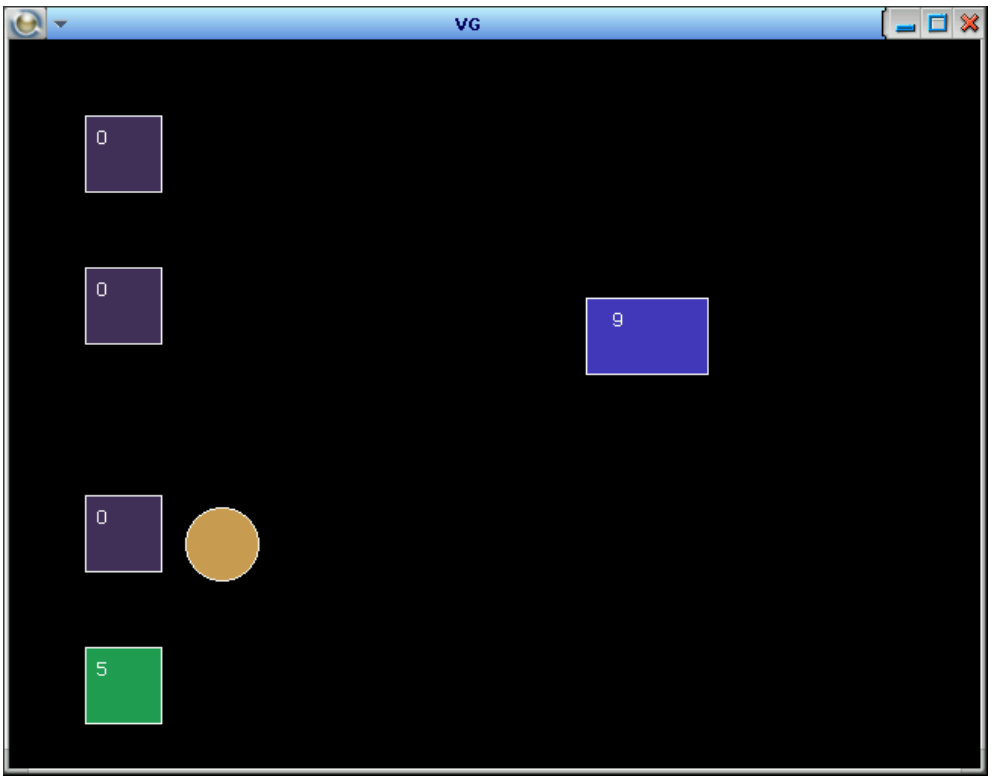

Рис.2 – Процесс работы программы с одним транспортным потоком

```
#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 chid = ChannelCreate(0);
int chid_2 = ChannelCreate(0);

int graphid = ConnectGraph();

void *boilerRoom(void * arg) {
   int coid = ConnectAttach(0, 0, chid, 0, 0);
   int numStation = * (int *) arg;
   setprio(0, numStation);
   sched_yield();
   int boiler = 0;
   int mark = 0;
   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 Switch Boiler-------------------------\n");
   }
   tPoint pos = GetPos(boiler);
   bool impuls = false;
   int text = Text(pos.x + 5, pos.y + 5, "0\0");
   while(true) {
```

```
    int text = Text(pos.x + 5, pos.y + 5, "0\0");
    while(true) {
        Fill(boiler, RGB(70, 50, 90));
        MsgSend(coid, &numStation, sizeof(int), &mark, sizeof(int));
        Fill(boiler, RGB(34,156, 87));
        char Temp[3] = {};
        while(mark > 0) {
            sprintf(Temp, "%d", mark);
            SetText(text, Temp);
            usleep(500000);
            mark--;
        }
        SetText(text, "0\0");
    }
}

void moveTo(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);
    }
```

```
        Move(car, dx, dy);
        usleep(10000);
    }
}

void *fuelTankThread(void *arg) {
    int coid = ConnectAttach(0, 0, chid_2, 0, 0);
    int fuelTank = Rect(380, 170, 80, 50);
    Fill(fuelTank, RGB(67,56, 187));
    int text = Text(395, 175, "0\0");
    int mark = 0;
    int ready = 0;
    char Temp[3] = {};
    while(true) {
        mark = 1 + rand() % 10;
        sprintf(Temp,"%d", mark);
        SetText(text, Temp);
        int rcvid = MsgReceive(chid_2, 0, sizeof(int), 0);
        MsgReply(rcvid, 0, &mark, sizeof(int));
    }
}

void *carThread(void *arg) {
    int car =  Ellipse(300, 170,50,50);
    Fill(car, RGB(200, 156, 87));
    int point = 0;
    int boiler = 0;
    int rcvid = 0;
    int t = 0;
    int coid = ConnectAttach(0, 0, chid, 0, 0);
    int coid_2 = ConnectAttach(0, 0, chid_2, 0, 0);
    while(true) {
        rcvid = MsgReceive(chid, &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 Switch-----------------------------\n");
```

```
            boiler = boiler1;
            break;
         case 2:
            boiler = boiler2;
            break;
         case 3:
            boiler = boiler3;
            break;
         case 4:
            boiler = boiler4;
            break;
         default:
            printf("-------------------------Error Switch-----------------------------\n");
      }
      int mark = 0;
      if(boiler != 0) {
         MsgSend(coid_2, 0, 0, &mark, sizeof(int));
         moveTo(boiler, car, true);
         MsgReply(rcvid, 0, &mark, sizeof(int));
         moveTo(boiler, car, false);
      }
      else
         printf("-------------------------Error Move-------------------------------\n");
   }
}

int main() {
   boiler1 = Rect(50,50,50,50);
   Fill(boiler1, RGB(34,156, 87));
   boiler2 = Rect(50,150,50,50);
   Fill(boiler2, RGB(34,156, 87));
   boiler3 = Rect(50,300,50,50);
   Fill(boiler3, RGB(34,156, 87));
   boiler4 = Rect(50,400,50,50);
   Fill(boiler4, RGB(34,156, 87));
   int tid = 0;
   int num_1 =1;
   pthread_create(0, 0, boilerRoom, &num_1);
   int num_2 = 2;
   pthread_create(0, 0, boilerRoom, &num_2);
   int num_3 = 3;
   pthread_create(0, 0, boilerRoom, &num_3);
   int num_4 = 4;
   pthread_create(0, 0, boilerRoom, &num_4);
   pthread_create(0, 0, fuelTankThread, 0);
   pthread_create(0, 0, carThread, 0);
   InputChar();
   CloseGraph();
}
```

2. А теперь добавьте второе транспортное средство.

Второе задание отличается от первого лишь добавлением дополнительного потока с транспортом.

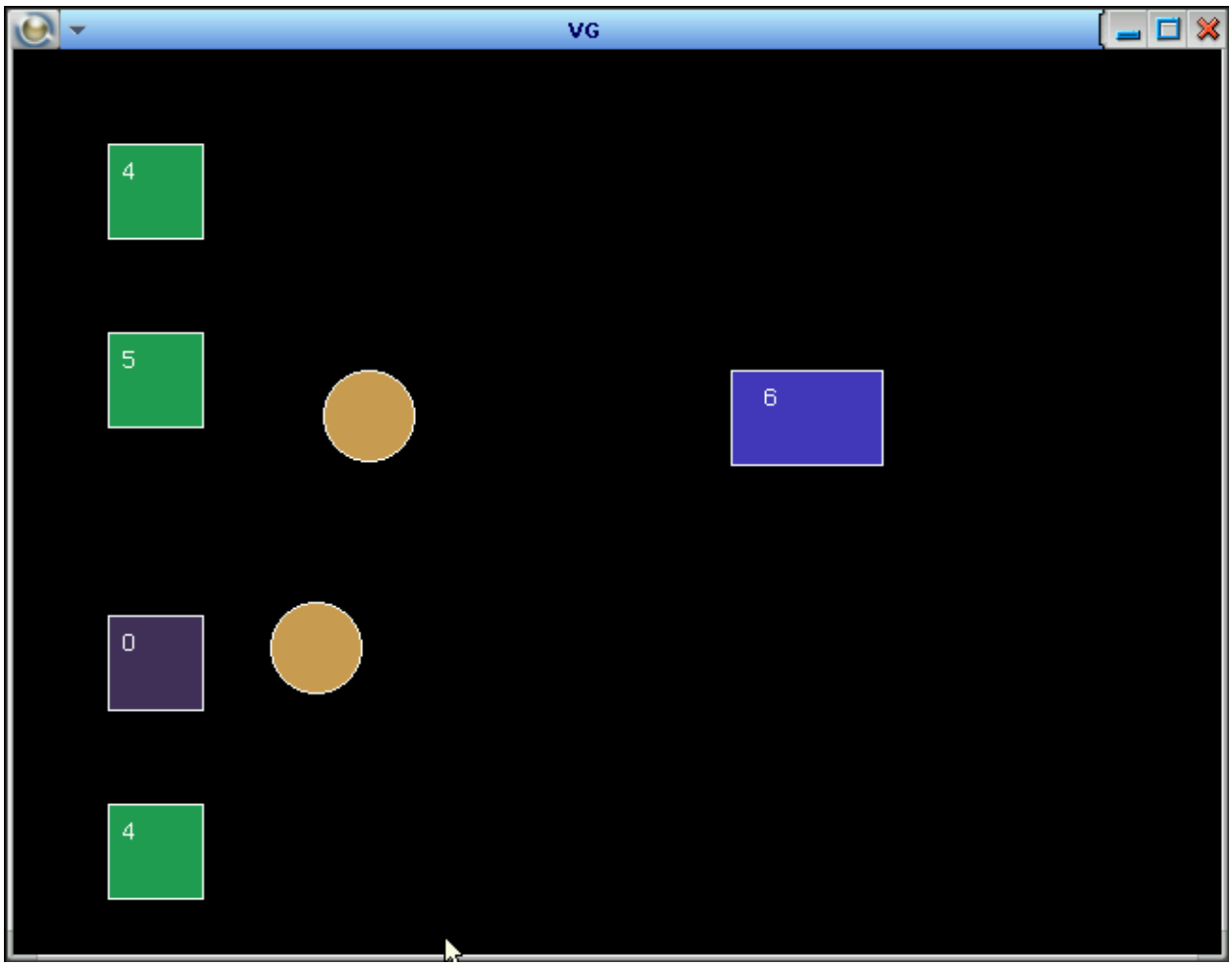

Рис.3 – Процесс работы программы с двумя транспортными потоком

```
int main() {
    boiler1 = Rect(50,50,50,50);
    Fill(boiler1, RGB(34,156, 87));
    boiler2 = Rect(50,150,50,50);
    Fill(boiler2, RGB(34,156, 87));
    boiler3 = Rect(50,300,50,50);
    Fill(boiler3, RGB(34,156, 87));
    boiler4 = Rect(50,400,50,50);
    Fill(boiler4, RGB(34,156, 87));
    int tid = 0;
    int num_1 =1;
    pthread_create(0, 0, boilerRoom, &num_1);
    int num_2 = 2;
    pthread_create(0, 0, boilerRoom, &num_2);
    int num_3 = 3;
    pthread_create(0, 0, boilerRoom, &num_3);
    int num_4 = 4;
    pthread_create(0, 0, boilerRoom, &num_4);
    pthread_create(0, 0, fuelTankThread, 0);
    pthread_create(0, 0, carThread, 0);
    pthread_create(0, 0, carThread, 0);
    InputChar();
    CloseGraph();
    return 0;
}
```

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.

Программа реализована с использованием pulse-методов. Они оповещают потоки о необходимости поставки.

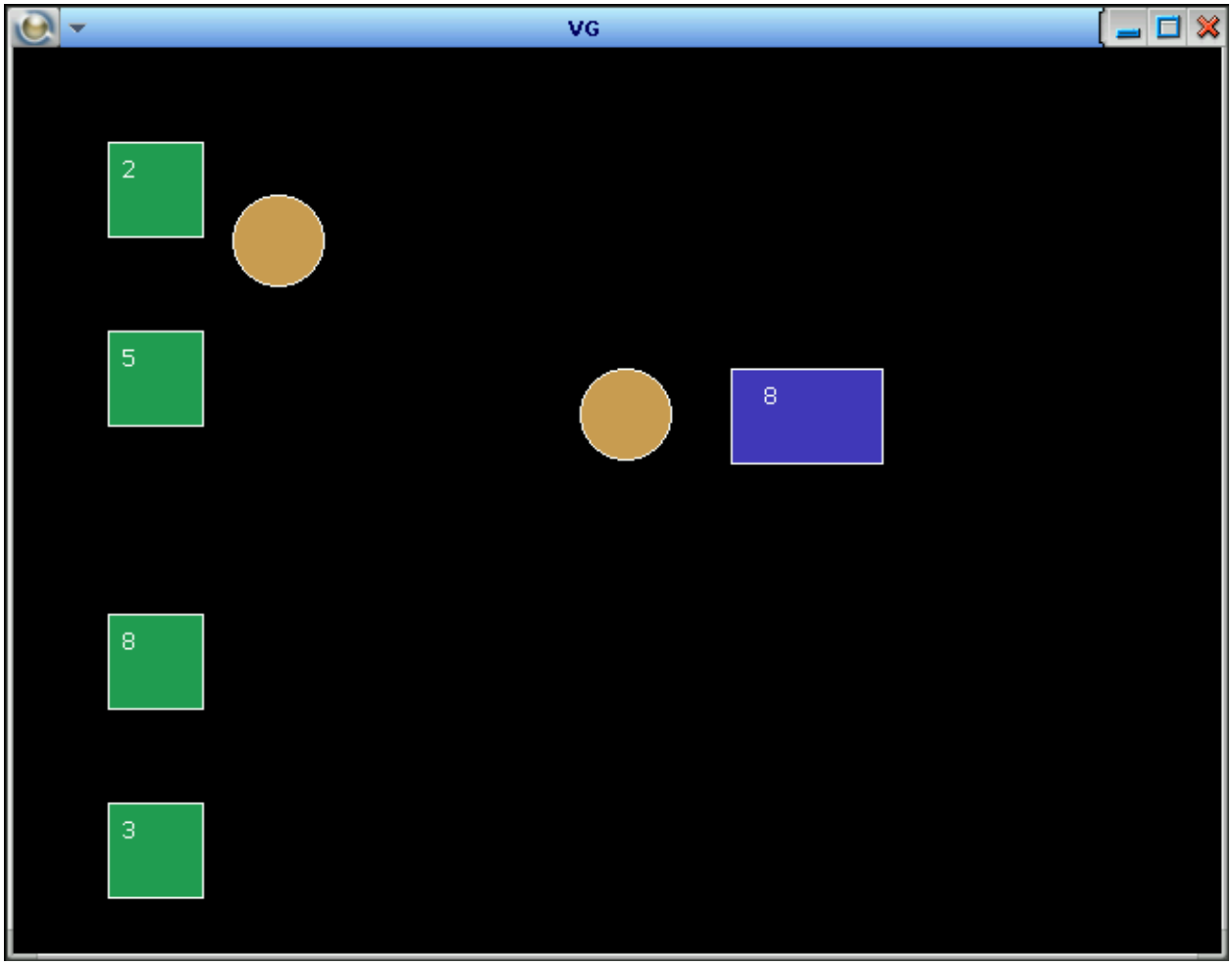

Рис. 4 – Процесс работы программы с импульсами (транспорт двигается к котлу с топливом < 3)

MsgSendPulse(int coid, int priority, int code, int value) – отправляет импульс к потоку.

```
while(true) {
    impuls = false;
    Fill(boiler, RGB(70, 50, 90));
    MsgSend(coid, &numStation, sizeof(int), &mark, sizeof(int));
    Fill(boiler, RGB(34,156, 87));
    char Temp[3] = { };
    while(mark > 0) {
        sprintf(Temp, "%d", mark);
        SetText(text, Temp);
        if(mark <= 3 && impuls != true) {
            MsgSendPulse(coid, numStation, 0, numStation);
            impuls = true;
        }
        usleep(500000);
        mark--;
    }
    SetText(text, "0\0");
}
```

```
    int mark = 0;

    if(boiler != 0)
    {
        MsgSend(coid_2, 0, 0, &mark, sizeof(int));
        moveTo(boiler, car, true);
        if(rcvid)
            MsgReply(rcvid, 0, &mark, sizeof(int));
        else
            MsgSendPulse(coid, mark, 0, mark);
        moveTo(boiler, car, false);
    }
    else
        printf("----------------------------Error Move--------------------------------\n");
}
```