Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

# Лабораторная работа №3

# «Электростанция»

Выполнил: студент 4 курса

ИВТ, гр. ИП-813

Бурдуковский И.А.

Выполнил с: Стояк Ю.К.

Проверил:

преподаватель кафедры ПМиК

Белевцова Екатерина Андреевна

Новосибирск, 2021 г.

# Оглавление

# Задание на лабораторную работу

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.
2. А теперь добавьте второе транспортное средство.
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
4. А теперь сделайте сетевой вариант разработанной программы. Пусть теперь хранилище работает на одной машине, котлы на другой, а транспортные средства переносят топливо между этими двумя машинами. Вам понадобится разделить вашу программу на две, которые вы будете запускать на разных узлах сети. Чтобы не надоедать соседу с просьбами о запуске вашей программки, которая к тому же будет постоянно подвешивать его компьютер, отладку можно вести на своем компьютере. Напомним, что имя узла содержится в переменной HOSTNAME.

# Выполнение

**1)** При выполнении задания были задействованы нити (потоки) из библиотеки <pthread> 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.

**2)** Второе транспортное средство легко добавить через добавление дополнительного потока с машиной в функции main() с помощью повторения вызова функции - pthread_create(0, 0, transport, $num[i]);

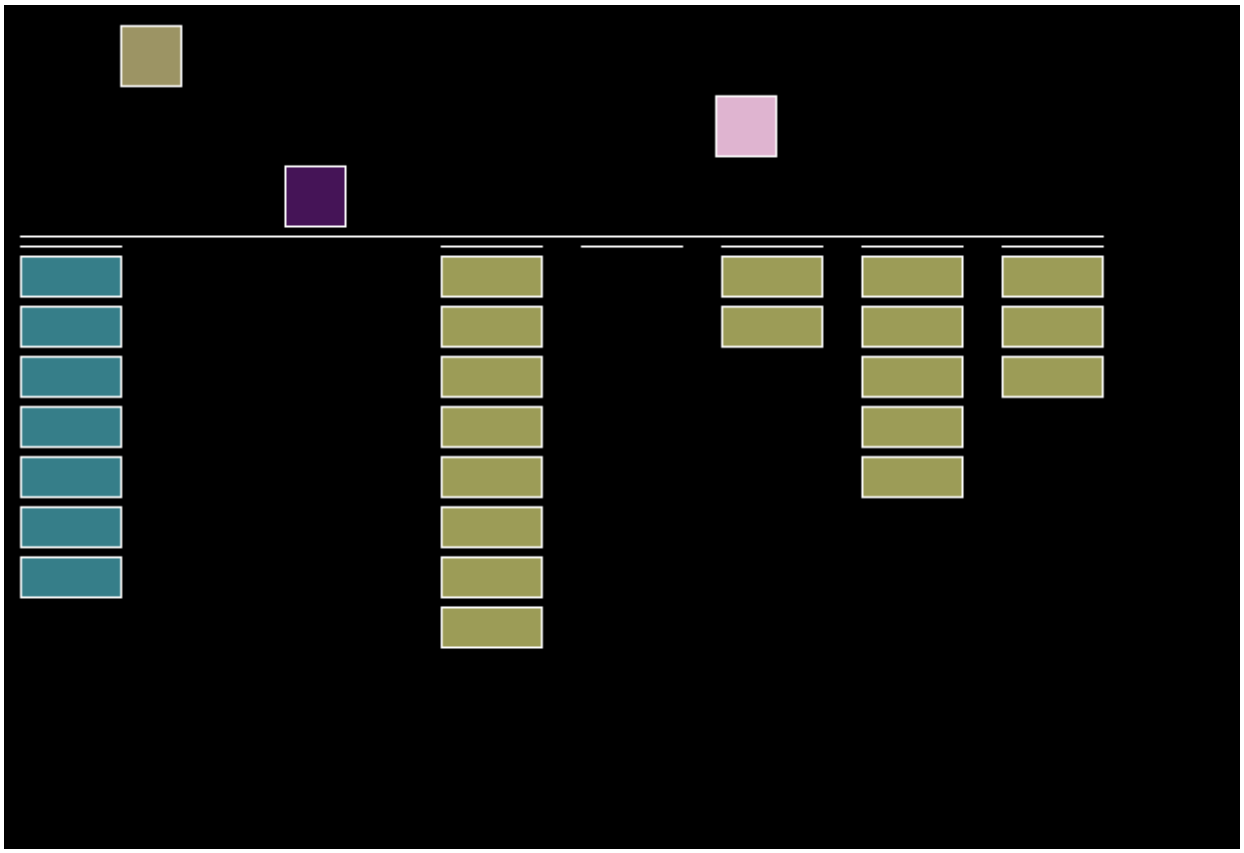
**3)** Для реализации данного задания были использованы pulse-методы, которые оповещают потоки о необходимости доставки для котлов у которых почти закончилось топливо. Чтобы отправить пульс используется функция MsgSendPulse(int coid, int priority, int code, int value) – отправляет импульс к потоку.

**4)** Для реализации этого задания предыдущая лабораторная была разбита на 2 файла – в одном были реализованные функции для транспорта и станции, а в другом функция для котлов. Взаимодействие этих файлов было реализовано через использование сети Qnet.

Запуск программы: в одном окне терминала (ttyp0) запускается файл, где хранится функция транспорта и станции. (cc file1.cpp –l vg –l m –o file1) В

другом окне терминала(ttyp1) запускается файл с функций котлов (cc file2.cpp –l vg –l m –o file2).

**Результат работы программы:**



# Код программы

## File1.cpp:

```
#include <stdlib.h>
#include <unistd.h>
#include <vingraph.h>
#include <iostream>
#include <stdio.h>
#include <sys/types.h>
#include <pthread.h>
#include <vector>
#include <time.h>
#include <sys/neutrino.h>
#include <sched.h>
#include <errno.h>
#include <sys/stat.h>
#include <fcntl.h>

#define OFFSET 35

using namespace std;

const int full = 10;
const int m = 5;
const int n = 3;

int channel1 = ChannelCreate(0);
int channel2 = ChannelCreate(0);
int cb[m];
int pid = getpid();
int pb;

void *transport(void *args)
{
    int offset = *((int*)args);
    int id = Rect(15,10+offset*OFFSET,30,30);
    int x = 20;
    int colo = RGB(rand()% 255, rand()% 255,rand()% 255);
    Fill(id,colo);
```

```
    int toX;
    int i = 0;
    int speed = 1;
    int mark=0;

    int buffer[4];

    int bid=0;
    int tmp;
    int coid = ConnectAttach(0,pid,channel1,0,0);
    int coidB[m];
    for(int i=0;i<m;i++){
        coidB[i]=ConnectAttach(0,pb,cb[i],0,0);
    }
    while(1){
        tmp=MsgReceive(channel2,&buffer,sizeof(buffer),0);

        if(tmp==0)
            bid=buffer[2];
        else
            bid=buffer[0];
        MsgSend(coid,0,0,&mark,sizeof(int));
        toX=220+70*bid;

        while(1){
            //Fill(id,RGB(255,0,0));
            Move(id,speed,0);
            usleep(10000);
            tPoint k=GetPos(id);
            if(k.x>=toX+x/2){
                MoveTo(toX+x/2,k.y,id);
                break;
            }
        }
        int a;
```

```
        if(tmp==0)
            MsgSend(coidB[bid],&mark,sizeof(mark),&a,sizeof(int));
        else
            if(tmp>0)
                MsgReply(tmp,0,&mark,sizeof(int));
        usleep(50000*(mark+1));
        while(1){
            Move(id,-speed,0);
            usleep(10000);
            tPoint k=GetPos(id);
            if(k.x<=x){
                MoveTo(x,k.y,id);
                break;
            }
        }
    }
}

void *fuelThread(void *args){
    srand(time(0));
    int count=0;
    int mark=rand()%7+4;
    vector<int> a;
    int tmp;
    Line(10, 10+OFFSET*n, 220+70*m-20, 10+OFFSET*n);
    Line(10, 15+OFFSET*n, 10+50, 15+OFFSET*n);
    while(1){
        if(count<mark){
            int id=Rect(10,20+OFFSET*n+count*25,50,20);
            a.push_back(id);
            Fill(id,RGB(54,126,137));
            count++;
        }
        else{
            tmp=MsgReceive(channel1,0,0,0);
```

```
             for(int i=0;i<mark;i++){
                 Delete(a.back());
                 a.pop_back();
                 usleep(50000);
             }
             MsgReply(tmp,0,&mark,sizeof(int));
             count=0;
             mark=rand()%7+4;
         }
         usleep(100000);
    }
}
int main()
{
    ConnectGraph();
    int fd=open("/dev/shmem/NAME",O_CREAT+O_RDWR,0666);
    write(fd,&pid,4);
    write(fd,&channel1,sizeof(int));
    write(fd,&channel2,sizeof(int));
    int num[n];
    for(int i=0;i<n;i++){
        num[i]=i;
    }
    int tmp=MsgReceive(channel1,&pb,sizeof(int),0);
    MsgReply(tmp,0,0,0);
    tmp=MsgReceive(channel1,cb,sizeof(int)*m,0);
    MsgReply(tmp,0,0,0);
    pthread_create(0,0,fuelThread,0);

    for(int i=0;i<n;i++){
        pthread_create(0,0,transport, &num[i]);
    }
    InputChar();
    CloseGraph();
    return 0;
}
```

**File2.cpp:**

```
#include <stdlib.h>
#include <unistd.h>
#include <vingraph.h>
#include <iostream>
#include <stdio.h>
#include <sys/types.h>
#include <pthread.h>
#include <vector>
#include <time.h>
#include <sys/neutrino.h>
#include <sched.h>
#include <errno.h>
#include <sys/stat.h>
#include <fcntl.h>

#define OFFSET 35

using namespace std;

const int full = 10;
const int m = 5;
const int n = 3;

int channel1;
int channel2;
int cb[m];
int pid;
int pb;

void *boilerThread(void *args){
    int num = *((int*)args);
    int buffer[4];
    buffer[0]=num;
    int mark=0;
    int markPulse=0;
    bool error=false;
```

```
    vector<int> a;
    int coid=ConnectAttach(0,pid,channel2,0,0);
    int pulseID=0;
    Line(220+70*num,15+OFFSET*n,220+70*num+50,15+OFFSET*n);

    while(MsgSend(coid,buffer,sizeof(buffer),&mark,sizeof(int))){
        usleep(500000);
    }
    while(1){
        if(pulseID==-1)
            MsgSend(coid,&buffer,sizeof(buffer),&mark,sizeof(int));

        for(int j=0;j<mark;j++){
            int id=Rect(220+70*num,20+OFFSET*n+j*25,50,20);
            Fill(id,RGB(156,156,87));
            a.push_back(id);
            usleep(50000);
        }
        while(mark>0){
            if(mark==2){
                pulseID=MsgSendPulse(coid,10,_PULSE_CODE_MINAVAIL+1+num,num);
            }
            sleep(1);
            mark--;
            Delete(a.back());
            a.pop_back();
        }
        if(pulseID==-1)
            continue;
        int tmp=MsgReceive(cb[num],&mark,sizeof(mark),0);
        MsgReply(tmp,0,0,0);
        usleep(10000);
    }
}
```

```
int main()
{
    ConnectGraph();

    pb=getpid();

    int fd=open("/dev/shmem/NAME",O_RDWR);
    read(fd,&pid,sizeof(int));
    read(fd,&channel1,sizeof(int));
    read(fd,&channel2,sizeof(int));
    int num[m];
    for(int i=0;i<m;i++){
        num[i]=i;
    }
    for(int i=0;i<m;i++){
        cb[i]=ChannelCreate(0);
    }
    int coid=ConnectAttach(0,pid,channel1,0,0);
    MsgSend(coid,&pb,sizeof(int),0,0);
    MsgSend(coid,cb,sizeof(int)*m,0,0);
    for(int i=0;i<m;i++){
        pthread_create(0,0,boilerThread,&num[i]);
    }
    InputChar();
    CloseGraph();
    return 0;
}
```

# Вывод

В этой лабораторной работе мы узнали о взаимодействии между нитями с помощью сообщений, применили знания об импульсах и сетях на практике.