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

# Лабораторная работа №4. Часть №1,2,3

# «Настоящая задача реального времени»

Выполнил: студент 4 курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: старший преподаватель кафедры ПМиК

Милешко А.В.

Новосибирск, 2020 г.

## Задание

**Ступень 1. Написать программу, сбивающую одну тарелку с помощью ракеты (тарелка движется слева направо).**
**Ступень 2. Написать программу, сбивающую несколько тарелок с помощью ракет (тарелки движутся в разных направлениях).**
**Ступень 3. Написать программу, сбивающую медленные тарелки ракетами, а быстрые -- управляемыми снарядами.**

## Выполнение задания

Для выполнения ступени № 1 необходимо определить подходящее время для запуска снаряда. Зная время пролета тарелки между двумя радарами, мы можем вычислить время, за которое тарелка достигнет середины экрана, где и стоит наше «ПВО». Также не стоит забывать, что ракета должна успеть долететь с Земли до цели – необходимо брать упреждение, взяв необходимое время нам нужно вычесть из времени полета тарелки время полета нашего снаряда.

Все временные отметки можно получить с помощью функции clock_gettime(clockid_t clock_id, struct timespec *t_time), где clock_id – идентификатор часов, а t_time – структура, в которую будет записано определенное время. Мы будем использовать реальное время, поэтому в качестве clock_id будем использовать флаг CLOCK_REALTIME. Чтобы произвести залп необходимо использовать регистр RG_GUNS, для точности задействуем залп из 15 выстрелов. Сложность выбрана 1, размер цели 3 - ни один воробей не пострадал :).


Рис.1 - Цель поражена.

Для ступени №2 основной цикл был вынесен в две схожие нити (потока) из библиотеки pthread – в одной шел опрос правого локатора (номер локатора – 4), а во второй левого (номер локатора – 1) для своевременного обнаружения тарелок по обоим сторонам. Сложность была выбрана 2 для того, чтобы тарелки летели с двух сторон.

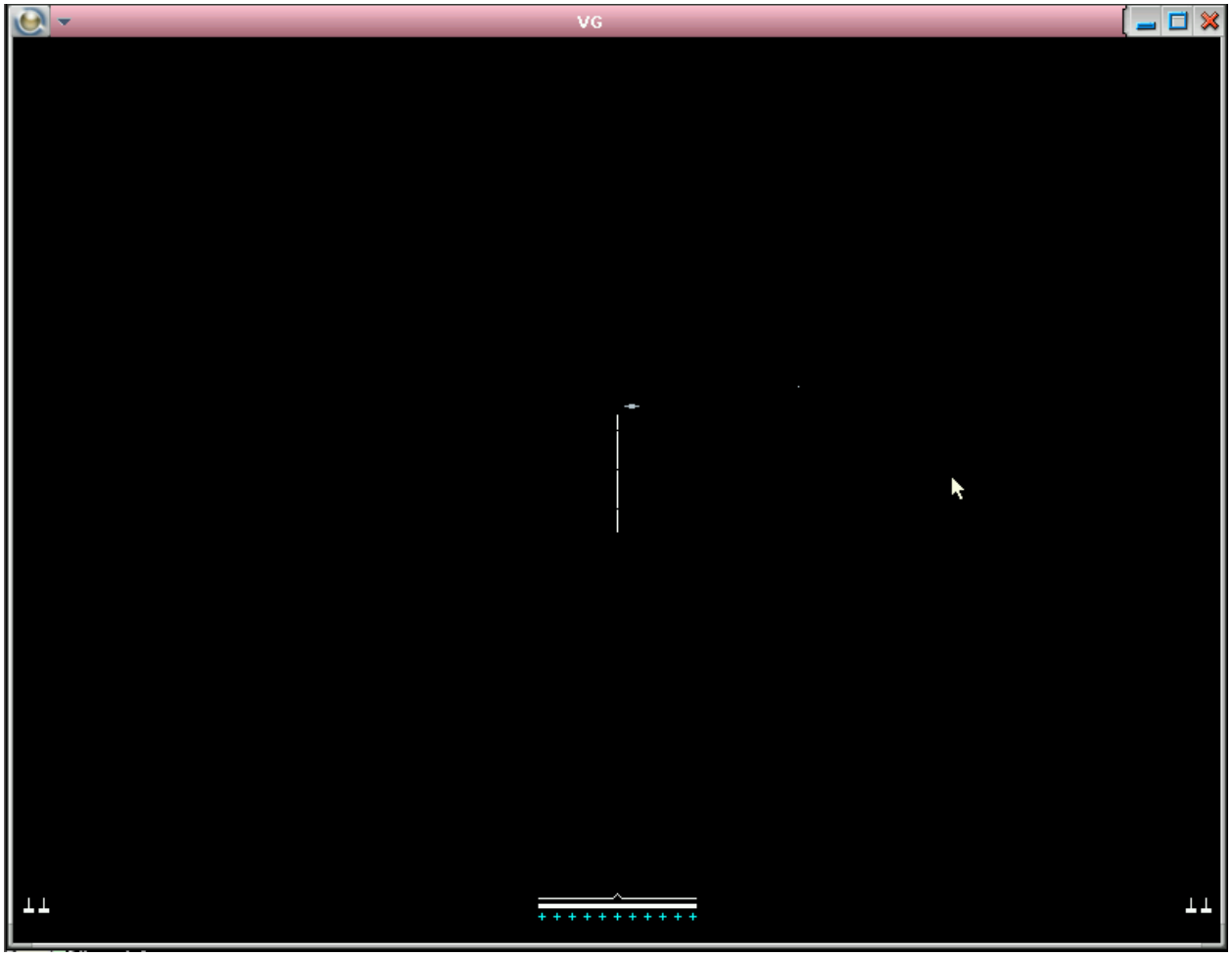


Рис.2 – залп летит поражать тарелку, летящую слева.

Для ступени №3, на основе программы для ступени №2, была реализована стрельба управляемыми снарядами через специальную переменную ammo, которая находится в разделяемой памяти (shared memory) для обозначения номера, выпускаемого снаряда, такая реализация обусловлена тем, чтобы оба потока могли обратиться к этой переменной.

Чтобы управляемый снаряд был выпущен, была добавлена проверка, что ракета не успеет долететь – это когда время до выстрела менее 200мс. Чтобы произвести стрельбу применяются изменения в регистрах RG_RCMN – выбор снаряда, RG_RCMC – направления полета и в зависимости от функции снаряд будет лететь либо влево, либо вправо. Сложность была оставлена №2, размер цели для проверки 3, опять же стрельба по воробьям не производится.


Рис.3 – Одна цель уже была поражена залпом и управляема ракета полетела уничтожать вторую цель и, это еще на скриншоте не произошло, но уверяю, что она сделала это успешно.

## Финальный листинг программы:

```
#include <sys/neutrino.h>
#include <unistd.h>
#include <vingraph.h>
#include <time.h>
#include <pthread.h>
#include <sys/mman.h>
#include "/root/labs/plates.h"

static int *ammo;

void *right_locator_thread(void *args)
{
        struct timespec start, stop;
        while(1)
        {
                usleep(1);
                if(getreg(RG_LOCN) == 4 && getreg(RG_LOCW) == 3)
                {
                        clock_gettime(CLOCK_REALTIME, &start);
                        while(1)
                        {
                                usleep(1);
                                if(getreg(RG_LOCN) == 3)
                                {
                                        clock_gettime(CLOCK_REALTIME, &stop);
```

```
                                        break;
                                }
                        }
                        long nsecs = stop.tv_nsec - start.tv_nsec;
                        long secs = stop.tv_sec - start.tv_sec;
                        if(secs > 0 && nsecs < 0)
                        {
                                nsecs += 1000000000;
                                secs--;
                        }
                        else if(secs < 0 && nsecs > 0)
                        {
                                nsecs -= 1000000000;
                                secs++;
                        }
                        double usec = (secs * 1000000000 + nsecs) / 1000.0;
                        int HEIGHT = getreg(RG_LOCY);
                        double target_speed = 10.0 / usec;
                        double target_center = 380.0 / target_speed;
                        double shot_time = ((570 - getreg(RG_LOCY)) / 100.0) *
1000000;
                        double shot_fire = target_center - shot_time - 300000;
                        if(usec <= 60000||shot_fire > 1100000000||shot_fire <
0.3)
                        {
                                if(*ammo <= 9)
                                {
                                        long flight_time = 4000 * (570 -
HEIGHT);
                                        putreg(RG_RCMN, *ammo);
                                        putreg(RG_RCMC, RCMC_START);
                                        usleep(flight_time);
                                        putreg(RG_RCMC, RCMC_RIGHT);
                                        *ammo += 1;
                                }
                        }
                        else
                        {
                                usleep(shot_fire);
                                for(int i = 0; i < 15; i++)
                                {
                                        usleep(50000);
                                        putreg(RG_GUNS, GUNS_SHOOT);
                                }
                        }
                }
        }
}

void *left_locator_thread(void *args)
{
        struct timespec start, stop;
        while(1)
        {
                usleep(1);
                if(getreg(RG_LOCN) == 1 && getreg(RG_LOCW) == 3)
                {
                        clock_gettime(CLOCK_REALTIME, &start);
                        while(1)
                        {
                                usleep(1);
                                if(getreg(RG_LOCN) == 2)
                                {
                                        clock_gettime(CLOCK_REALTIME, &stop);
```

```
                                        break;
                                }
                        }
                        long nsecs = stop.tv_nsec - start.tv_nsec;
                        long secs = stop.tv_sec - start.tv_sec;
                        if(secs > 0 && nsecs < 0)
                        {
                                nsecs += 1000000000;
                                secs--;
                        }
                        else if(secs < 0 && nsecs > 0)
                        {
                                nsecs -= 1000000000;
                                secs++;
                        }
                        double usec = (secs * 1000000000 + nsecs) / 1000.0;
                        int HEIGHT = getreg(RG_LOCY);
                        double target_speed = 10.0 / usec;
                        double target_center = 380.0 / target_speed;
                        double shot_time = ((570 - getreg(RG_LOCY)) / 100.0) *
1000000;
                        double shot_fire = target_center - shot_time - 300000;
                        if(usec <= 60000||shot_fire > 1100000000||shot_fire <
0.3)
                        {
                                if(*ammo <= 9)
                                {
                                        long flight_time = 4000 * (570 -
HEIGHT);
                                        putreg(RG_RCMN, *ammo);
                                        putreg(RG_RCMC, RCMC_START);
                                        usleep(flight_time);
                                        putreg(RG_RCMC, RCMC_LEFT);
                                        *ammo += 1;
                                }
                        }
                        else
                        {
                                usleep(shot_fire);
                                for(int i = 0; i < 15; i++)
                                {
                                        usleep(50000);
                                        putreg(RG_GUNS, GUNS_SHOOT);
                                }
                        }
                }
        }
}

int main()
{
        ammo = static_cast<int*>(mmap(NULL, sizeof *ammo,
PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, -1, 0));
        *ammo = 0;
        pthread_t left_locator, right_locator;
        StartGame(2);
        pthread_create(&left_locator, NULL, &left_locator_thread, NULL);
        pthread_create(&right_locator, NULL, &right_locator_thread, NULL);
        pthread_join(left_locator, 0);
        pthread_join(right_locator, 0);
        EndGame();
        return 0;
}
```