**Министерство науки и высшего образования**

**Российской федерации**

#### ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

##### **Кафедра «Математическое обеспечение и применение ЭВМ»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| «Утверждаю» | | | | |
| Зав. кафедрой «МО и ПЭВМ» | | | | |
|  | | | Макарычев П.П. | |
| « » |  |  | | 2019 |

### Пояснительная записка

к курсовой работе по дисциплине

«Операционные системы»

на тему: «Моделирование системы управления железнодорожным переездом»

|  |  |
| --- | --- |
| Автор работы: | Угроватов Д.В. |
| Направление бакалавриата | 09.03.04 («Программная инженерия») |
| Обозначение курсовой работы | ПГУ 09.03.04 - 04КР161.24 ПЗ |
| Группа | 16ВП1 |
| Руководитель работы | Шашков Б.Д., к.т.н., профессор |
| Работа защищена «\_\_» \_\_\_\_ 2019 г. | Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Пенза 2019 г.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Реферат**  Пояснительная записка содержит 28 листов, 9 рисунков, 3 таблицы, 4 использованных источников, 1 приложение.  ОПЕРАЦИОННЫЙ СИСТЕМЫ, UNIX, ПРИЛОЖЕНИЯ С ПАРАЛЛЕЛЬНЫМИ ПРОЦЕССАМИ, СИГНАЛЫ  Объектом исследования являются средства управления параллельными процессами в UNIX.  Цель работы – разработка программы, моделирующей систему управления железнодорожным переездом.  Разработка проводилась на языке программирования C++ в среде программирования Code::blocks.  Осуществлено функциональное тестирование разработанного программного обеспечения, которое показало корректность его работы. | | | | | | | | | | |
|  |  |  |  |  | ПГУ 09.03.04 - 06КР161.24 ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Угроватов Д.В. |  |  | «Моделирование системы управления железнодорожным переездом»  Пояснительная записка. | Лит. | | | Лист | Листов |
| Пров. | | Шашков Б.Д. |  |  |  |  |  | 3 | 28 |
|  | |  |  |  | Группа 16ВП1 | | | | |
| Н. контр. | |  |  |  |
| Утв. | |  |  |  |

**Оглавление**

[Введение 5](#_Toc9879110)

[1. Разработка системы управления железнодорожным переездом 6](#_Toc9879111)

[1.1. Постановка задачи 6](#_Toc9879112)

[1.2. Анализ требований 7](#_Toc9879113)

[1.2.2. Нефункциональные требования 9](#_Toc9879114)

[1.3. Проектирование 10](#_Toc9879115)

[1.3.1. Проектирование структур и функций 10](#_Toc9879116)

[1.3.2. Проектирование параллельных вычислений 11](#_Toc9879117)

[1.3.3. Проектирование интерфейса 12](#_Toc9879118)

[1.4. Тестирование программы 13](#_Toc9879119)

[1.5. Тестирование 14](#_Toc9879121)

[Заключение 17](#_Toc9879123)

[Список используемой литературы 18](#_Toc9879124)

[Приложение А – исходный код программы 19](#_Toc9879125)

# Введение

Параллельные вычисления – одна из самых актуальных тем в сфере IT технологий. Данный термин означает способ организации компьютерных вычислений, при котором программы разрабатываться как набор взаимодействующих вычислительных процессов, работающих параллельно.

Основная сложность при проектировании параллельных программ – обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами.

В настоящее время современные операционные системы представляют множество различных средств для создания программ с параллельными вычислениями. Например, операционные системы семейства Linux.

Linux – семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU. Особенностью операционной системы является поддержка стандартов POSIX.

UNIX – семейство переносимых многозадачных и многопользовательских операционных систем, которые основаны на идеях оригинального проекта AT&T UNIX.

POSIX (Portable Operating System Interface – переносимый интерфейс операционных систем) – набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой, библиотеку языка C и набор приложений и их интерфейсов.

**Цель работы:** создать программу, моделирующую систему управления железнодорожным переездом.

В рамках данной работы будут пройдены следующие этапы:

* Составлены требования к программе
* Спроектирована программа
* Реализована программа
* Протестирована программа
* Сделаны выводы

# 1. Разработка системы управления железнодорожным переездом

### 1.1. Постановка задачи

В рамках курсовой работы необходимо разработать программу, моделирующую работу железнодорожного переезда.

Система управления железнодорожным переездом обеспечивает безопасность движения автотранспорта при пересечении железнодорожной магистрали. При приближении поезда переезд должен быть закрыт (шлагбаум опущен). Ж/д переезд может быть доступен или полностью закрыт для движения автотранспорта. Если переезд находится в работоспособном состоянии, то управление осуществляется либо в ручном, либо в автоматическом режимах. В ручном режиме дежурный лично отвечает за своевременное закрытие и открытие движения автотранспорта через переезд. В автоматическом режиме переезд закрывается при приближении поезда на заданное расстояние и открывается при отдалении поезда на заданное расстояние. В обоих режимах имеется возможность подачи звукового и светового сигнала при приближении поезда. Дежурный может задать параметры автоматического управления: значение расстояния, на которое должен приблизиться поезд, чтобы переезд закрылся; значение расстояния, на которое должен удалиться поезд, чтобы переезд открылся; необходимость подачи звукового или светового сигнала.

Способ обмена информацией между параллельными процессами – сигналы.

Разработка программы должна быть выполнена на языке C или C++.

Программное обеспечение должно быть полностью отлажено и протестировано, функционировать под управлением ОС семейства Linux.

Требования к интерфейсу отсутствуют.

### 1.2. Анализ требований

#### 1.2.1. Функциональные требования

В ходе анализа задания на разработку были выделены следующие варианты использования (Рисунок 1).

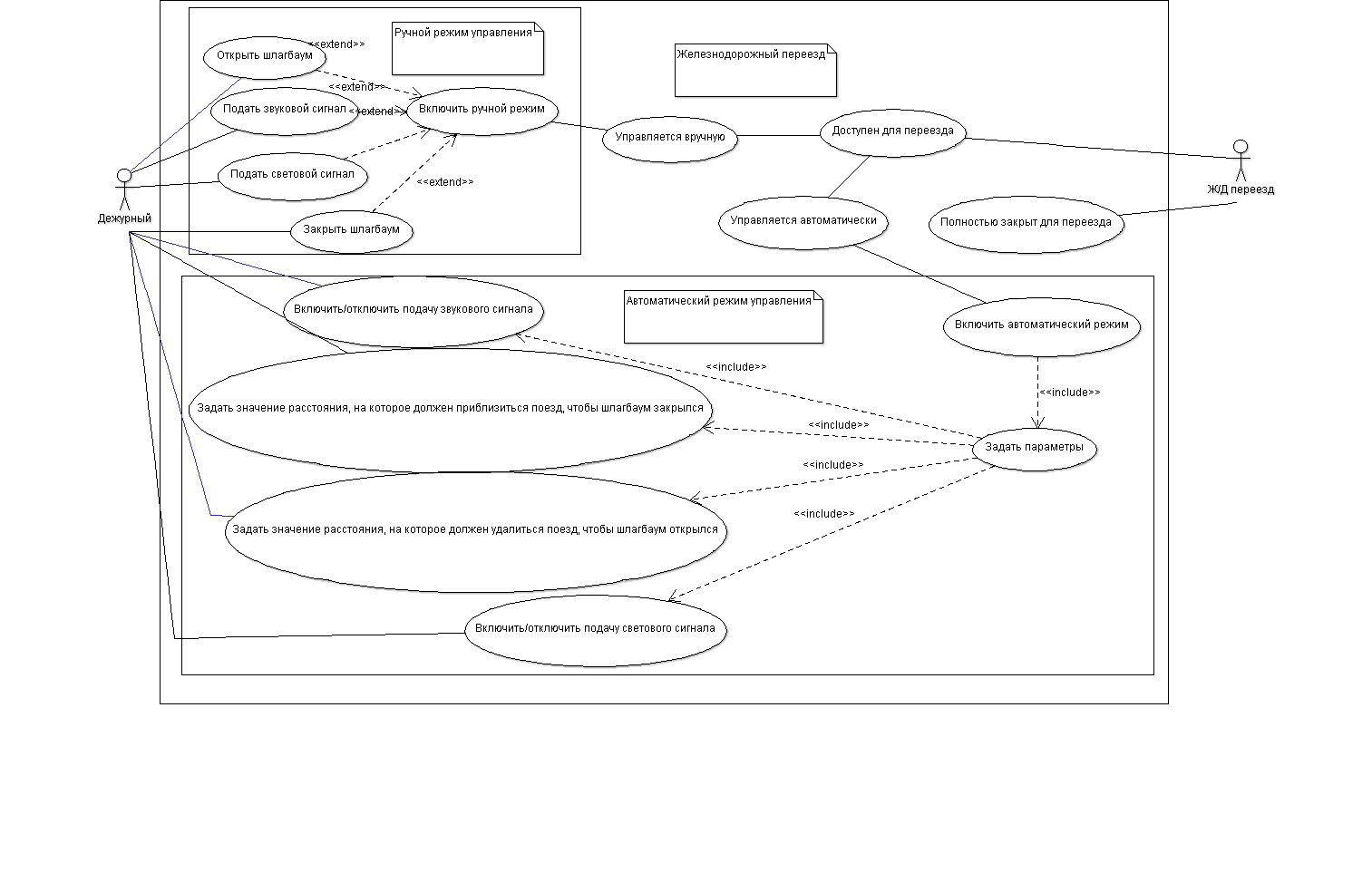


Рисунок 1 – Диаграмма вариантов использования

Опишем спецификацию нескольких прецедентов (Таблицы 1-2).

Таблица 1 - спецификация прецедента «Открыть шлагбаум»

|  |
| --- |
| Прецедент: Открыть шлагбаум |
| ID: 1 |
| Краткое описание: Включить ручной режим управления и открыть шлагбаум. |
| Действующие лица: Пользователь |
| Предусловие: Запущена программа |
| Основной поток:   1. ЕСЛИ переезд закрыт    1. Пользователь вводит команду для открытия переезда    2. Система передает команду в необходимый поток и выполняет команду    3. Система выводит на экран результат выполнения команды и меню выбора режима управления 2. Пользователь вводит команду для перехода в ручной режим 3. Система передает команду в необходимый поток и выполняет команду 4. Система выводит на экран результат выполнения команды и меню выбора действий в ручном режиме управления 5. Пользователь вводит команду для открытия шлагбаума 6. Система передает команду в необходимый поток и выполняет команду 7. Система выводит на экран результат выполнения команды и меню выбора действий в ручном режиме управления |
| Постусловие: нет |

Таблица 2 - спецификация прецедента «Включить автоматический режим»

|  |
| --- |
| Прецедент: Включить автоматический режим |
| ID: 2 |
| Краткое описание: Включить автоматический режим управления и задать параметры работы. |
| Действующие лица: Пользователь |
| Предусловие: Запущена программа |
| Основной поток:   1. ЕСЛИ переезд закрыт    1. Пользователь вводит команду для открытия переезда    2. Система передает команду в необходимый поток и выполняет команду    3. Система выводит на экран результат выполнения команды и меню выбора режима управления 2. Пользователь вводит команду для перехода в автоматический режим 3. Система передает команду в необходимый поток и выполняет команду 4. Система выводит на экран результат выполнения команды и меню ввода параметров 5. Пользователь вводит, нужно ли подавать звуковой сигнал 6. Пользователь вводит, нужно ли подавать световой сигнал 7. Пользователь вводит, на какое расстояние должен отдалиться поезд, чтобы шлагбаум открылся 8. Пользователь вводит, на какое расстояние должен приблизиться поезд, чтобы шлагбаум закрылся 9. Система передает параметры в необходимый поток и выполняет команду |
| Постусловие: нет |

### 1.2.2. Нефункциональные требования

Одним из нефункциональных требований является то, что система должна быть разбита на две подпрограммы. Одна подпрограмма будет выполнять роль «терминала», то есть из нее будет происходить управление железнодорожным переездом. Вторая же будет «монитором», то есть она будет выводить результат выполнения команд, полученных от «терминала». Также в ней будет храниться информация о текущем состоянии переезда.

Таким образом в будущей программе необходимо реализовать достаточно много параллельных вычислений. Одной из важных задач при проектировании такой программы является необходимость обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами. Для решения данной задачи сформулировано еще одно нефункциональное требование – «способ обмена информацией между параллельными процессами – сигналы».

### 1.3. Проектирование

### 1.3.1. Проектирование структур и функций

Для проектирования структуры была выделена следующая сущность: переезд.

Структура:

struct pereezd {

/\*\*

\* открыт ли шлагбаум

\*/

bool BarierOpened;

/\*\*

\* открыт ли переезд

\*/

bool PereezdOpened;

/\*\*

\* режим управления переездом

\*/

PereezdMode Status;

/\*\*

\* нужно ли делать звуковой сигнал при приближении поезда

\*/

bool SoundSignal;

/\*\*

\* нужно ли делать световой сигнал при приближении поезда

\*/

bool LightSignal;

/\*\*

\* расстояние, на которое должен отдалиться поезд, чтобы шлагбаум закрылся

\*/

int OpendDist;

/\*\*

\* расстояние, на которое должен приблизиться поезд, чтобы шлагбаум открылся

\*/

int CloseDist;

};

Перечисление PereezdStatus:

enum PereezdMode {

None=0,

Manual=1,

Automaic=2

};

Функции:

* Открыть/закрыть переезд
* Открыть/закрыть шлагбаум
* Подать световой сигнал
* Подать звуковой сигнал
* Включить ручной режим управления
* Включить автоматический режим

### 1.3.2. Проектирование параллельных вычислений

Для моделирования работы системы управления железнодорожным переездом было решено разделить «Терминал» и «Монитор» на разные процессы. Данный выбор связан с тем, что такое разделение лучше всего продемонстрирует работу заданной системы. Процесс «Терминал» будет отвечать за управление переездом, а процесс «Монитор» будет выводить информацию о проделанных действиях, состоянии переезда и другие данные на экран.

Таким образом система будет состоять из двух процессов. Для обмена информацией будут использоваться сигналы, а в роли буфера выступит файл.

### 1.3.3. Проектирование интерфейса

Поскольку требования к интерфейсу не были предъявлены, то можно спроектировать любой интерфейс, позволяющий использовать все функции приложения.

Основной задачей интерфейса является передача команд, которые вводит пользователь, в систему для их исполнения. Для данной задачи наиболее простым и удобным решением является реализация консольного командного интерфейса.

Таким образом, пользовательский интерфейс для «Монитора» будет представлять собой консольное окно для вывода информации, а для «Терминала» окно условно будет разделено на две части – область меню и область для ввода команд (Рисунок 2).

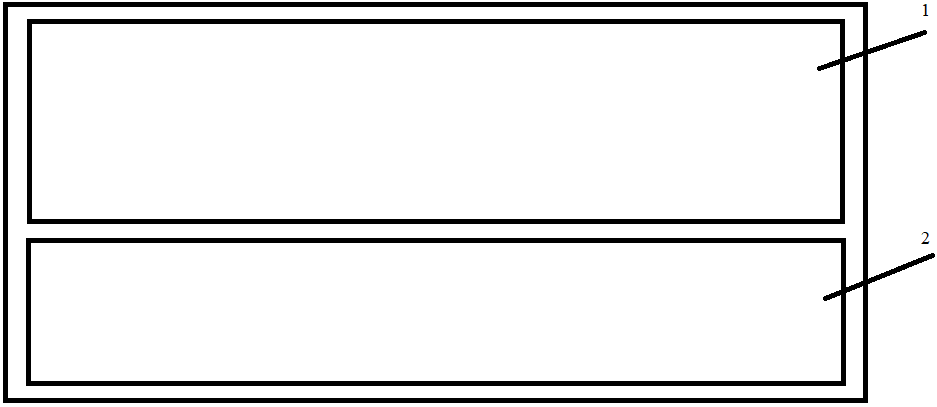


Рисунок 2 – интерфейс «Терминала»

1. Область для вывода меню
2. Область для ввода команд

## 1.4. Тестирование программы

### В соответствии с поставленными требованиями, на основе результатов проектирования была реализована программа (Рисунок 3).

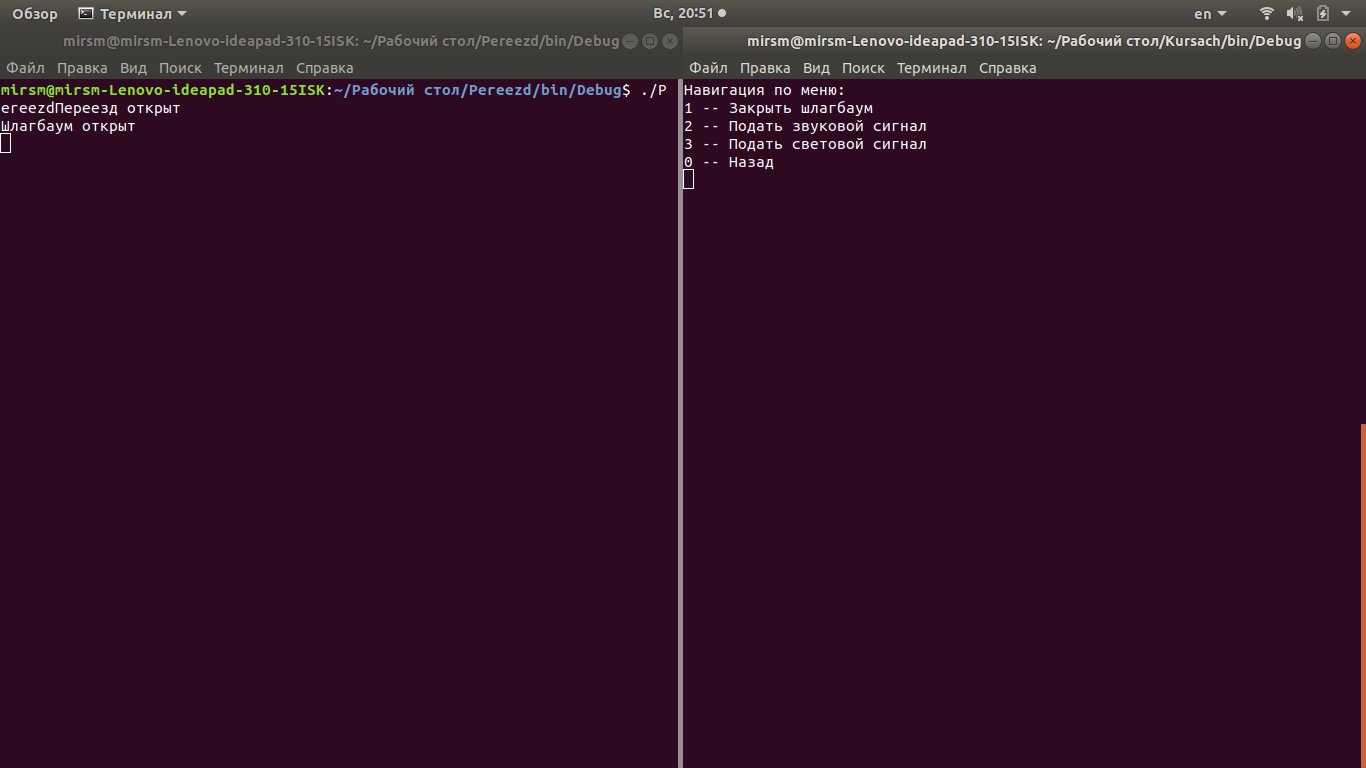
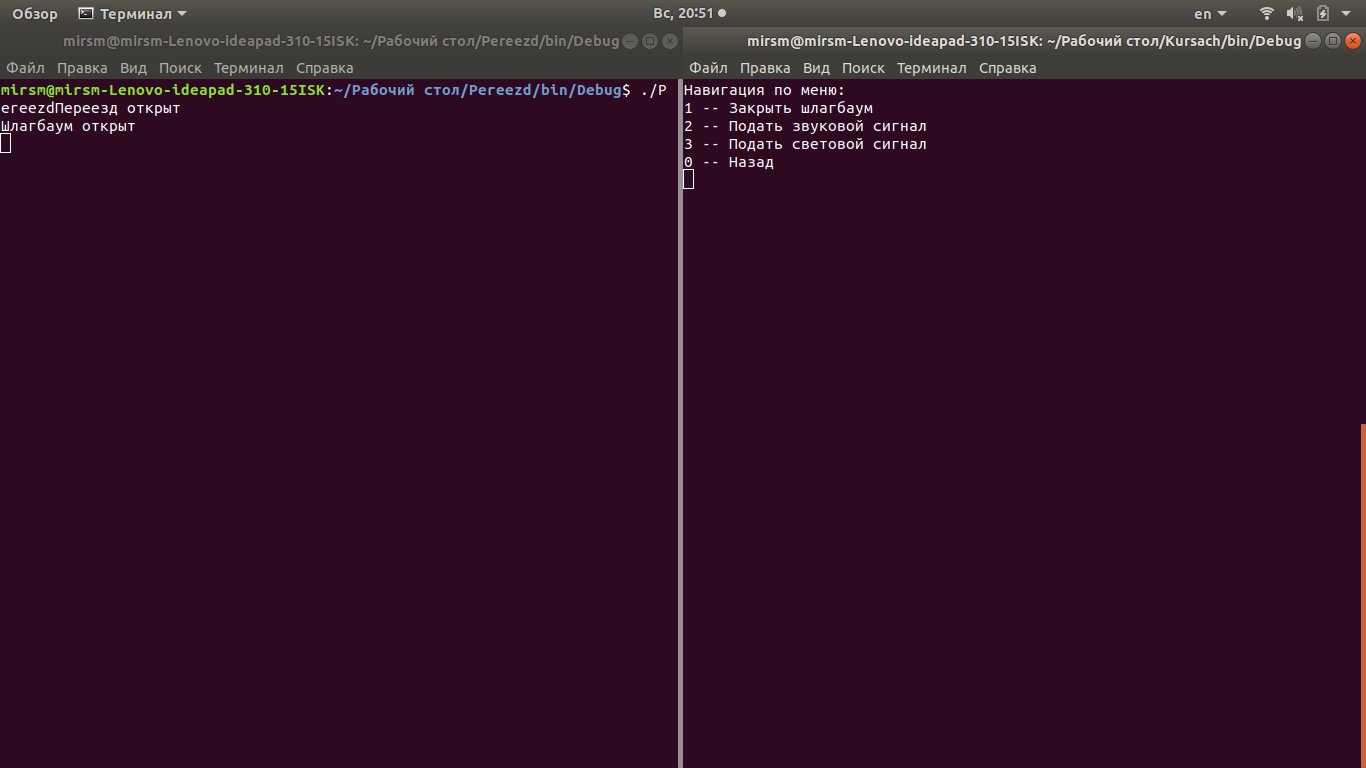


Рисунок 3 – результат запуска программы

Код программы представлен в приложении А. Описание работы функций представлено в коде.

### 1.5. Тестирование

Для проверки работоспособности системы было проведено функциональное тестирование. В таблице 3 представлены функциональные тесты.

Таблица 3 – функциональные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер теста** | **Название теста** | **Параметры теста** | **Результат** |
| 1 | Открыть шлагбаум | ----------------- | Тест пройден (Рисунок 4) |
| 2 | Подать световой сигнал | ----------------- | Тест пройден (Рисунок 5) |
| 3 | Включить автоматический режим | Подача звукового сигнала – включена;  Подача светового сигнала – выключена;  Расстояние до открытия – 350;  Расстояние до закрытия – 450; | Тест пройден (Рисунок 6) |
| 4 | Запустить поезд | Подача звукового сигнала – включена;  Подача светового сигнала – выключена;  Расстояние до открытия – 350;  Расстояние до закрытия – 450; | Тест пройден (Рисунок 7) |
| 5 | Вывести состояние поезда | ----------------- | Тест пройден (Рисунок 8) |
| 6 | Закрыть переезд | ----------------- | Тест пройден (Рисунок 9) |

## D:\16вп1\6 семестр\ОС\Kursach\Снимок экрана от 2019-05-19 20-51-18.png

Рисунок 4 – результат теста 1

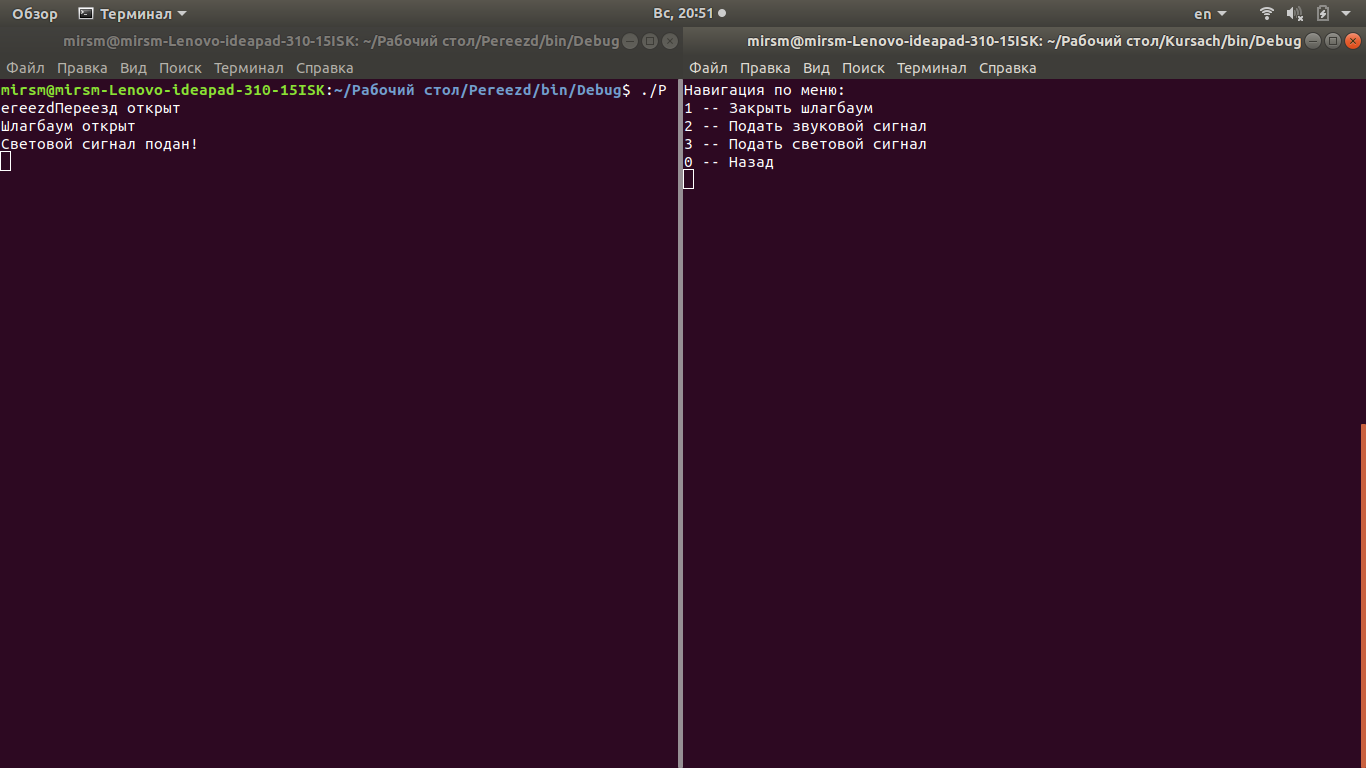


Рисунок 5 – результат теста 2

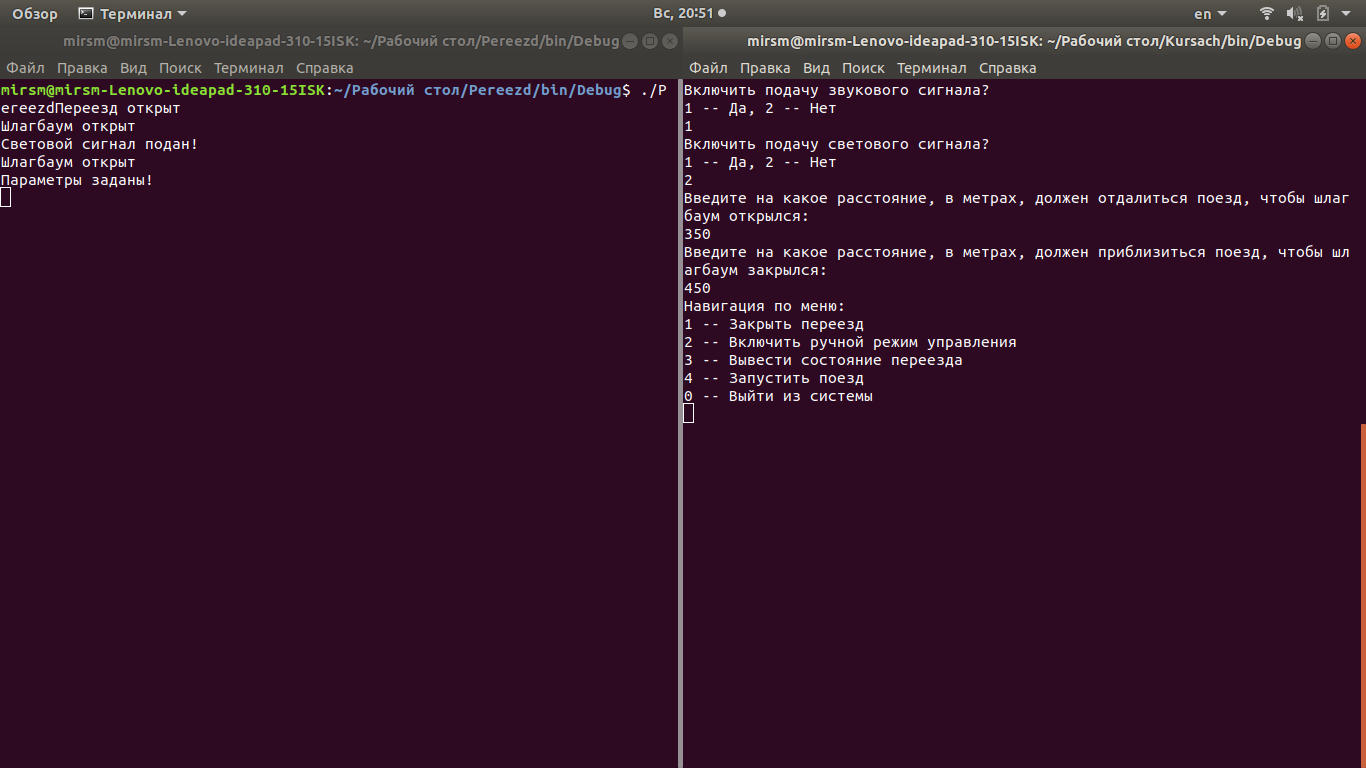


Рисунок 6 - результат теста 3

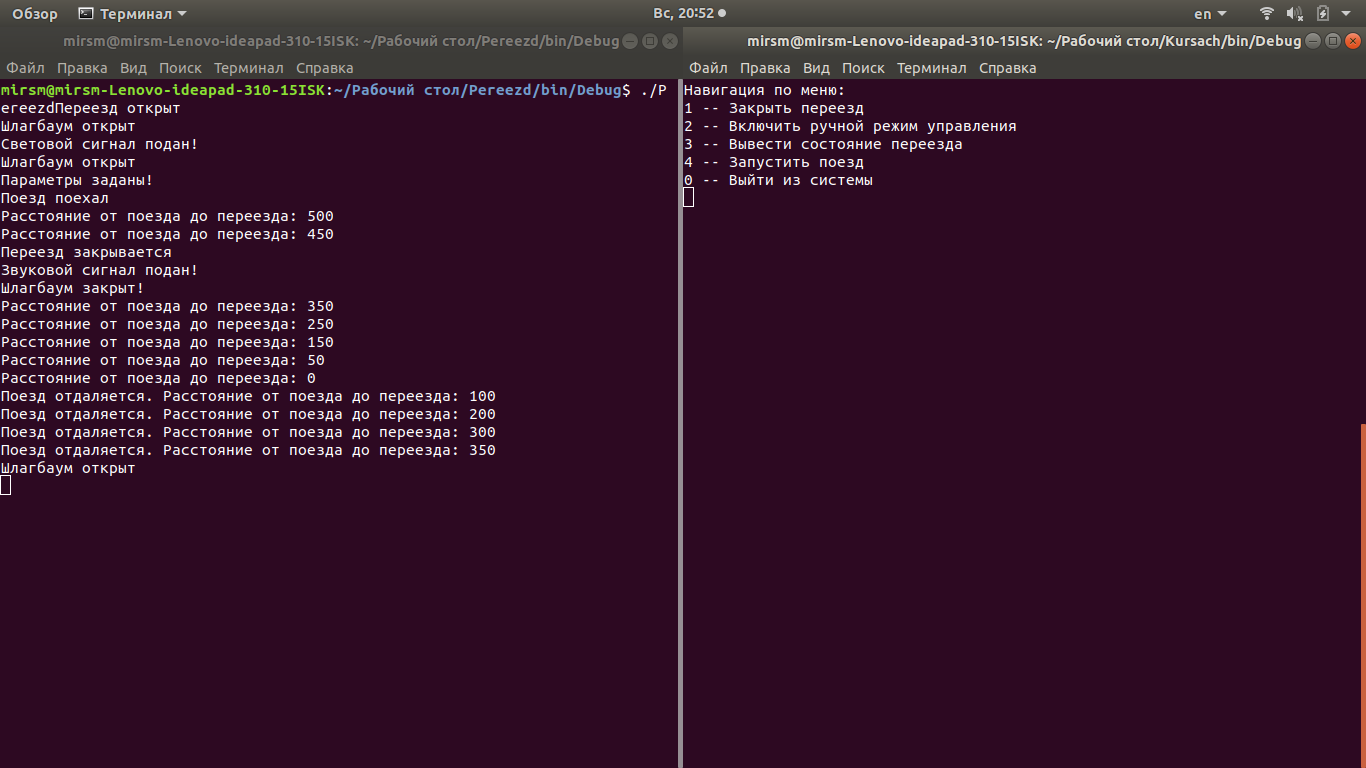


Рисунок 7 - результат теста 4

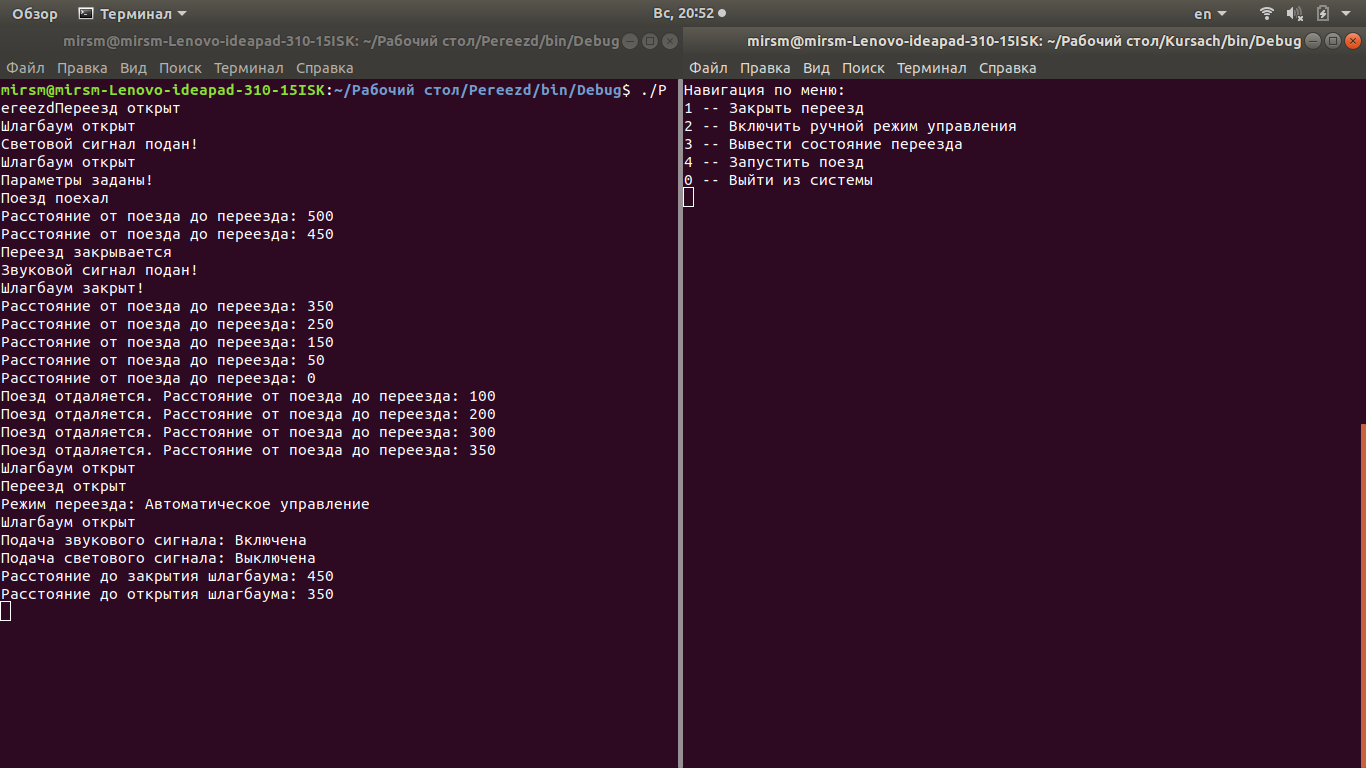


Рисунок 8 - результат теста 5

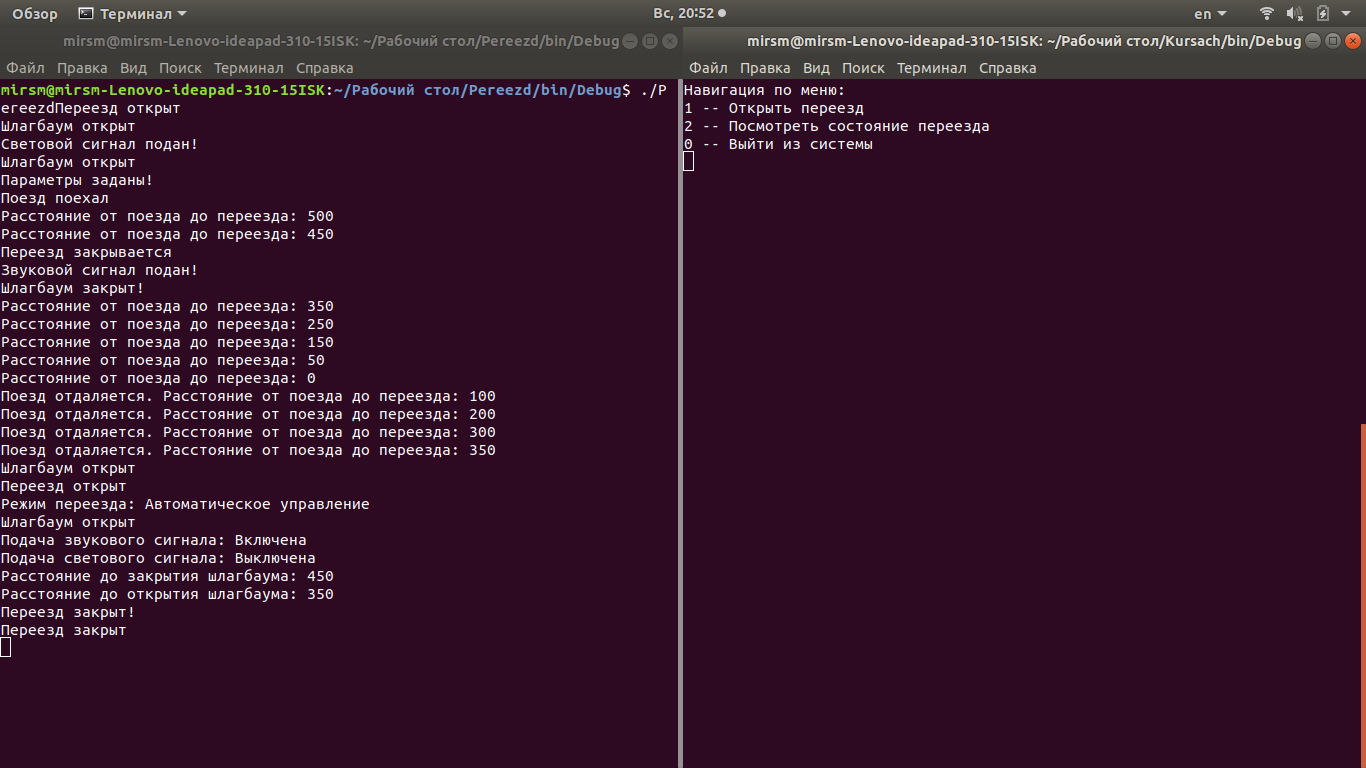


Рисунок 9 - результат теста 6

# Заключение

В рамках данной курсовой работы были выполнены все поставленные задачи:

• Составлены требования к программе

• Спроектирована программа

• Реализована программа

• Протестирована программа

Во время выполнения курсового проекта были изучены способы создания и синхронизации параллельных вычислений, функции для работы с сигналами в Unix-подобных системах, принципы работы в ОС семейства Linux.

Была создана программа на языке C++ для моделирования работы железнодорожного переезда.

Созданная программ позволяет выполнять следующие функции:

• Запускать моделирование железнодорожного переезда

• Изменять параметры системы (переключать режимы управления, изменять параметры для автоматического режима)

• Просматривать состояние системы при помощи ввода команд.

Созданная программа была протестирована, согласно поставленным требованиям. Тестирование показало корректность работы программы.

Полученное приложение удовлетворяет всем изложенным требованиям.

# Список используемой литературы

1. Шашков Б.Д. Методические указания к лабораторным работам по дисциплине «Операционные системы и сети» / Пенза, издательство ПГУ, 2013.
2. Брайан Уорд «Внутреннее устройство Linux» / Санкт-Петербург, издательство «Питер», 2016.
3. Мэтью Н., Стоунс Р. «Основы программирования в Linux» / Санкт-Петербург, издательство «БХВ-Петербург», 2009
4. Руководство по программированию на  
    C++ [Электронный ресурс]. – Режим доступа:   
   <https://metanit.com/cpp/tutorial/> (Дата обращения: 15.05.2019)

# Приложение А – исходный код программы

**Файл main.cpp («Монитор»):**

#include <iostream>

#include <signal.h>

#include <unistd.h>

#include <fstream>

using namespace std;

//Сигналы

#define SIGPZD 1

#define SIGMAN 2

#define SIGAUT 3

#define SIGCHK 4

#define SIGBAR 5

#define SIGTRN 6

#define SIGSND 7

#define SIGLGT 8

#define KEY1 49

#define KEY2 50

#define PEREEZDPIDFILE "/home/mirsm/Рабочий стол/Pereezd/PereezdPid"

#define CONSOLEPIDFILE "/home/mirsm/Рабочий стол/Kursach/ConsolePid"

#define PEREEZDPARAMS "/home/mirsm/Рабочий стол/Pereezd/PereezdParams"

enum PereezdMode

{

None=0,

Manual=1,

Automatic=2

};

struct pereezd {

bool BarierOpened;

bool PereezdOpened;

PereezdMode PereezdStatus;

bool SoundSignal;

bool LightSignal;

int DistToClose;

int DistToOpen;

};

pereezd Pereezd;

int ConsolePid;

void OpenOrCloseBarier(int sig);

void OpenOrClosePereezd(int sig);

void PrintStatus(int sig);

void ClearConsole();

void SetManualMode(int sig);

void SetAutomaticMode(int sig);

void StartTrain(int sig);

void MakeSoundSignal(int sig);

void MakeLightSignal(int sig);

int main()

{

ConsolePid=-1;

ofstream fout(PEREEZDPIDFILE);

fout<<getpid();

fout.close();

Pereezd.BarierOpened=false;

Pereezd.PereezdOpened=false;

Pereezd.PereezdStatus=PereezdMode::None;

Pereezd.SoundSignal=false;

Pereezd.LightSignal=false;

Pereezd.DistToClose=0;

Pereezd.DistToOpen=0;

signal(SIGBAR,OpenOrCloseBarier);

signal(SIGPZD,OpenOrClosePereezd);

signal(SIGCHK,PrintStatus);

signal(SIGTRN,StartTrain);

signal(SIGMAN,SetManualMode);

signal(SIGAUT,SetAutomaticMode);

signal(SIGSND,MakeSoundSignal);

signal(SIGLGT,MakeLightSignal);

while(1)

{

}

return 0;

}

/\*\*

\* открыть/закрыть шлагбаум

\* @param sig идентификатор сигнала

\*/

void OpenOrCloseBarier(int sig)

{

Pereezd.BarierOpened=!Pereezd.BarierOpened;

if(!Pereezd.BarierOpened)

{

cout<<"Шлагбаум закрыт!\n";

}

else

{

cout<<"Шлагбаум открыт\n";

}

}

/\*\*

\* открыть =/закрыть переезд

\* @param sig идентификатор сигнала

\*/

void OpenOrClosePereezd(int sig)

{

Pereezd.PereezdOpened=!Pereezd.PereezdOpened;

if(!Pereezd.PereezdOpened)

{

cout<<"Переезд закрыт!\n";

}

else

{

cout<<"Переезд открыт\n";

}

}

/\*\*

\* создать процессы для каждого резевуара

\* @param sig идентификатор сигнала

\*/

void PrintStatus(int sig)

{

if(Pereezd.PereezdOpened)

{

cout<<"Переезд открыт\n";

}

else

{

cout<<"Переезд закрыт\n";

return;

}

cout<<"Режим переезда: ";

switch(Pereezd.PereezdStatus)

{

case PereezdMode::None:

cout<<"Не установлен\n";

break;

case PereezdMode::Manual:

cout<<"Ручное управление\n";

if(Pereezd.BarierOpened)

cout<<"Шлагбаум открыт\n";

else

cout<<"Шлагбаум закрыт\n";

break;

case PereezdMode::Automatic:

cout<<"Автоматическое управление\n";

if(Pereezd.BarierOpened)

cout<<"Шлагбаум открыт\n";

else

cout<<"Шлагбаум закрыт\n";

cout<<"Подача звукового сигнала: ";

if(Pereezd.SoundSignal)

cout<<"Включена\n";

else

cout<<"Выключена\n";

cout<<"Подача светового сигнала: ";

if(Pereezd.LightSignal)

cout<<"Включена\n";

else

cout<<"Выключена\n";

cout<<"Расстояние до закрытия шлагбаума: "<<Pereezd.DistToClose<<endl;

cout<<"Расстояние до открытия шлагбаума: "<<Pereezd.DistToOpen<<endl;

break;

}

}

/\*\*

\* очистить консоль

\*/

void ClearConsole()

{

printf("\E[H\E[2J");

}

/\*\*

\* установить ручной режим управления

\* @param sig идентификатор сигнала

\*/

void SetManualMode(int sig)

{

Pereezd.PereezdStatus=PereezdMode::Manual;

}

/\*\*

\* установить параметры для автоматического режима

\* @param sig идентификатор сигнала

\*/

void SetAutomaticMode(int sig)

{

Pereezd.PereezdStatus=PereezdMode::Automatic;

Pereezd.BarierOpened=true;

cout<<"Шлагбаум открыт\n";

ifstream fin(PEREEZDPARAMS);

int nBuf;

fin>>nBuf;

if(nBuf==1)

Pereezd.SoundSignal=true;

else

Pereezd.SoundSignal=false;

fin>>nBuf;

if(nBuf==1)

Pereezd.LightSignal=true;

else

Pereezd.LightSignal=false;

fin>>Pereezd.DistToOpen;

fin>>Pereezd.DistToClose;

fin.close();

cout<<"Параметры заданы!\n";

}

/\*\*

\* запустить поезд

\* @param sig идентификатор сигнала

\*/

void StartTrain(int sig)

{

ifstream fin(CONSOLEPIDFILE);

fin>>ConsolePid;

fin.close();

cout<<"Поезд поехал\n";

int dist=500;

if(dist<Pereezd.DistToClose)

dist=Pereezd.DistToClose+100;

cout<<"Расстояние от поезда до переезда: "<<dist<<endl;

while(dist-100>=Pereezd.DistToClose)

{

dist-=100;

cout<<"Поезд приближается. Расстояние от поезда до переезда: "<<dist<<endl;

}

dist=Pereezd.DistToClose;

cout<<"Расстояние от поезда до переезда: "<<dist<<endl;

cout<<"Переезд закрывается\n";

if(Pereezd.SoundSignal)

MakeSoundSignal(1);

if(Pereezd.LightSignal)

MakeLightSignal(1);

OpenOrCloseBarier(1);

while(dist-100>0)

{

dist-=100;

cout<<"Расстояние от поезда до переезда: "<<dist<<endl;

}

dist=0;

cout<<"Расстояние от поезда до переезда: "<<dist<<endl;

while(dist+100<=Pereezd.DistToOpen)

{

dist+=100;

cout<<"Поезд отдаляется. Расстояние от поезда до переезда: "<<dist<<endl;

}

dist=Pereezd.DistToOpen;

cout<<"Поезд отдаляется. Расстояние от поезда до переезда: "<<dist<<endl;

OpenOrCloseBarier(1);

kill(ConsolePid,SIGABRT);

}

/\*\*

\* подать световой сигнал

\* @param sig идентификатор сигнала

\*/

void MakeLightSignal(int sig)

{

cout<<"Световой сигнал подан!\n";

}

/\*\*

\* подать звуковой сигнал

\* @param sig идентификатор сигнала

\*/

void MakeSoundSignal(int sig)

{

cout<<"Звуковой сигнал подан!\n";

}

**Файл main.cpp («Терминал»):**

#include <iostream>

#include <fcntl.h>

#include <signal.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <fstream>

using namespace std;

#define CH\_NAME "/home/mirsm/Рабочий стол/Pereezd/bin/Debug/Pereezd"

#define PEREEZDPIDFILE "/home/mirsm/Рабочий стол/Pereezd/PereezdPid"

#define CONSOLEPIDFILE "/home/mirsm/Рабочий стол/Kursach/ConsolePid"

#define PEREEZDPARAMS "/home/mirsm/Рабочий стол/Pereezd/PereezdParams"

//Нажатые клавиши

#define KEY0 48

#define KEY1 49

#define KEY2 50

#define KEY3 51

#define KEY4 52

//Сигналы

#define SIGPZD 1

#define SIGMAN 2

#define SIGAUT 3

#define SIGCHK 4

#define SIGBAR 5

#define SIGTRN 6

#define SIGSND 7

#define SIGLGT 8

enum PereezdMode

{

None=0,

Manual=1,

Automatic=2

};

bool PereezdOpened;

PereezdMode PereezdStatus;

bool BarierOpened;

bool SoundSignal;

bool LightSignal;

int DistToClose;

int DistToOpen;

int PereezdPid;

bool contin;

void CreateTrain();

void GetSignal(int sig);

void ManualMenu();

void ClearConsole();

void OpenOrCloseBarier();

void MakeSoundSignal();

void MakeLightSignal();

void OpenOrClosePereezd();

void SetAutomaticMode();

void PrintStatus();

void SetContin(int sig);

int main()

{

cout<<"Добро пожаловать в систему управления ж/д переездом!"<<endl;

contin=true;

PereezdOpened=false;

PereezdStatus=PereezdMode::None;

BarierOpened=false;

SoundSignal=false;

LightSignal=false;

DistToClose=0;

DistToOpen=0;

ofstream fout(CONSOLEPIDFILE);

fout<<getpid();

fout.close();

ifstream fin(PEREEZDPIDFILE);

fin>>PereezdPid;

fin.close();

signal(SIGABRT,SetContin);

int nChild=fork();

sleep(1);

while(1)

{

cout<<"Навигация по меню:\n";

if(!PereezdOpened)

{

cout<<"1 -- Открыть переезд\n2 -- Посмотреть состояние переезда\n0 -- Выйти из системы\n";

}

else

{

cout<<"1 -- Закрыть переезд\n";

if(PereezdStatus==PereezdMode::None)

{

cout<<"2 -- Включить ручной режим управления\n3 -- Включить автоматический режим управления\n";

cout<<"4 -- Посмотреть состояние переезда\n0 -- Выйти из системы\n";

}

if(PereezdStatus==PereezdMode::Manual)

{

cout<<"2 -- Вернуться к ручному управлению\n";

cout<<"3 -- Включить автоматический режим управления\n";

cout<<"4 -- Посмотреть состояние переезда\n0 -- Выйти из системы\n";

}

if(PereezdStatus==PereezdMode::Automatic)

{

cout<<"2 -- Включить ручной режим управления\n";

cout<<"3 -- Вывести состояние переезда\n4 -- Запустить поезд\n0 -- Выйти из системы\n";

}

}

int key=cin.get();

cin.get();

cout<<key<<endl;

ClearConsole();

switch(key)

{

case KEY0:

return 0;

case KEY1:

OpenOrClosePereezd();

break;

case KEY2:

if(PereezdOpened)

{

PereezdStatus=PereezdMode::Manual;

ManualMenu();

}

else

{

PrintStatus();

}

break;

case KEY3:

if(PereezdOpened)

{

if(PereezdStatus==PereezdMode::None ||PereezdStatus==PereezdMode::Manual)

{

SetAutomaticMode();

}

else

{

PrintStatus();

}

}

else

{

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

break;

case KEY4:

if(PereezdOpened && PereezdStatus!=PereezdMode::Automatic)

PrintStatus();

else

{

if(PereezdStatus==PereezdMode::Automatic)

{

if(!BarierOpened)

{

OpenOrCloseBarier();

}

CreateTrain();

}

else

{

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

}

break;

default:

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

break;

}

while(!contin)

{

}

}

return 0;

}

/\*\*

\* включить ручной режим управления

\*/

void ManualMenu()

{

kill(PereezdPid,SIGMAN);

while(1)

{

cout<<"Навигация по меню:\n";

if(BarierOpened)

cout<<"1 -- Закрыть шлагбаум\n";

else

cout<<"1 -- Открыть шлагбаум\n";

cout<<"2 -- Подать звуковой сигнал\n";

cout<<"3 -- Подать световой сигнал\n0 -- Назад\n";

int key=cin.get();

cin.get();

ClearConsole();

switch(key)

{

case KEY0:

return;

case KEY1:

OpenOrCloseBarier();

break;

case KEY2:

MakeSoundSignal();

break;

case KEY3:

MakeLightSignal();

break;

default:

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

break;

}

}

}

/\*\*

\* запустить поезд

\*/

void CreateTrain()

{

contin=false;

kill(PereezdPid,SIGTRN);

}

/\*\*

\* очистить консоль

\*/

void ClearConsole()

{

system("clear");

}

/\*\*

\* открыть/закрыть шлагбаум

\*/

void OpenOrCloseBarier()

{

BarierOpened=!BarierOpened;

kill(PereezdPid,SIGBAR);

}

/\*\*

\* подать звуковой сигнал

\*/

void MakeSoundSignal()

{

kill(PereezdPid,SIGSND);

}

/\*\*

\* подать световой сигнал

\*/

void MakeLightSignal()

{

kill(PereezdPid,SIGLGT);

}

/\*\*

\* открыть/закрыть переезд

\*/

void OpenOrClosePereezd()

{

kill(PereezdPid,SIGPZD);

if(PereezdOpened)

{

PereezdOpened=false;

PereezdStatus=PereezdMode::None;

}

else

{

PereezdOpened=true;

}

}

/\*\*

\* включить автоматический режим управления и задать параметры

\*/

void SetAutomaticMode()

{

PereezdStatus=PereezdMode::Automatic;

while(1)

{

cout<<"Включить подачу звукового сигнала?\n1 -- Да, 2 -- Нет\n";

int key=cin.get();

cin.get();

bool founded=false;

switch(key)

{

case KEY1:

SoundSignal=true;

founded=true;

break;

case KEY2:

SoundSignal=false;

founded=true;

break;

default:

founded=false;

break;

}

if(founded)

break;

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

while(1)

{

cout<<"Включить подачу светового сигнала?\n1 -- Да, 2 -- Нет\n";

int key=cin.get();

cin.get();

bool founded=false;

switch(key)

{

case KEY1:

LightSignal=true;

founded=true;

break;

case KEY2:

LightSignal=false;

founded=true;

break;

default:

founded=false;

break;

}

if(founded)

break;

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

while(1)

{

cout<<"Введите на какое расстояние, в метрах, должен отдалиться поезд, чтобы шлагбаум открылся:\n";

try

{

cin>>DistToOpen;

if(DistToOpen<0)

throw new exception();

break;

}catch(...){

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

}

while(1)

{

cout<<"Введите на какое расстояние, в метрах, должен приблизиться поезд, чтобы шлагбаум закрылся:\n";

try

{

cin>>DistToClose;

if(DistToClose<0)

throw new exception();

break;

}catch(...){

cout<<"Введен неверный символ! Пожалуйста, повторите попытку:\n";

}

}

ofstream fout(PEREEZDPARAMS);

cin.get();

if(SoundSignal)

fout<<"1 ";

else

fout<<"2 ";

if(LightSignal)

fout<<"1 ";

else

fout<<"2 ";

fout<<DistToOpen<<" ";

fout<<DistToClose<<" ";

fout.close();

BarierOpened=true;

kill(PereezdPid,SIGAUT);

}

/\*\*

\* вывести информацию о переезде

\*/

void PrintStatus()

{

kill(PereezdPid,SIGCHK);

sleep(1);

}

/\*\*

\* ожидать, пока поезд проедет

\*/

void SetContin(int sig)

{

contin=true;

}