Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | *Информационных технологий и компьютерных систем* |
|  |  |
| Кафедра | *Информатики и вычислительной техники* |
|  |  |

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| по дисциплине | ***Операционные системы*** |
|  |  |
| на тему | Многопоточня Linux программная модель транспортных перевозок дилижансами по круговому маршруту |

**Пояснительная записка**

|  |  |
| --- | --- |
| Шифр проекта |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Студента (ки) | | | | **Шкаброва Данилы Андреевича** | | | | | | |
|  |  |  |  | | | | фамилия, имя, отчество полностью | | | | | | |
|  |  |  | Курс | ***2*** |  | | | Группа | | **ПИ-182** | | |  |
|  |  |  |  |  | |  | | |  |  | | |  |
|  |  |  | Направление (специальность) | | | | | | | *09.03.03 –* | | | |
|  |  |  | *Прикладная информатика* | | | | | | | | | | |
|  |  |  | код, наименование | | | | | | | | | | |
|  |  |  | Руководитель | | | | *к.т.н., доцент* | | | | | | |
|  |  |  |  | | | | ученая степень, звание | | | | | | |
|  |  |  | *Флоренсов А.Н* | | | | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | | | | |
|  |  |  | Выполнил (а) | | | |  | | | | | | |
|  |  |  |  | | | | дата, подпись студента (ки) | | | | | | |
|  |  |  | К защите | | | |  | | | | | | |
|  |  |  |  | | | | дата, подпись руководителя | | | | | | |
|  |  |  | Выполнение и подготовка к защите КП (КР) | | | | | Защита КП (КР) | | | | Итоговый рейтинг | |
|  |  |  |  | | | | |  | | | |  | |
|  |  |  | Проект (работа) защищен (а) с оценкой | | | | | | | |  | | |

Омск 2020

Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

**ОТЗЫВ**

**на курсовой проект (работу)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Факультет | | | | | | | *Информационных технологий и компьютерных систем* | | | | | | | | | | |
|  | | | | | | |  | | | | | | | | | | |
| Кафедра | | | *Информатики и вычислительной техники* | | | | | | | | | | | | | | |
|  | | |  | | | | | | | | | | | | | | |
| Дисциплина | | | | ***Операционные системы*** | | | | | | | | | | | | | |
|  | | | |  | | | | | | | | | | | | | |
| Тема | Многопоточня Linux программная модель транспортных перевозок дилижансами по | | | | | | | | | | | | | | | | |
|  | | круговому маршруту | | | | | | | | | | | | | | | |
| Студент | | | **Шкабров Данила Андреевич** | | | | | | | | | | | | | | |
|  | | | фамилия, имя, отчество полностью | | | | | | | | | | | | | | |
| Курс | ***2*** | | | | Группа | | | **ПИ-**182 |  | | | | | | | | |
|  | |  | | |  | | |  |  | | | | | | | | |
| Руководитель | | | | | к.т.н., доцент Флоренсов А.Н. | | | | | | | | | | | | |
|  | | | | | ученая степень, звание, ФИО | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| **Содержание отзыва** | | | | | | | | | | | | | | | | | |
| Спроектирована укрупненная схема алгоритма для проекта реализации | | | | | | | | | | | | | | | | | |
| многопоточной программной модели транспортных перевозок дилижансами по | | | | | | | | | | | | | | | | | |
| круговому маршруту на Linux | | | | | | | | | | | | | | | | | |
| Детализированы и разработаны схемы алгоритмов для отдельных функций и | | | | | | | | | | | | | | | | | |
| подзадач. Разработана и отлажена программа, реализующая программную модель | | | | | | | | | | | | | | | | | |
| транспортных перевозок дилижансами по круговому маршруту. | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| **Рейтинговые баллы за выполнение и подготовку к защите курсового проекта (работы)** | | | | | | | | | | | | |  | | | |
| **Заключение о допуске к защите** | | | | | | | | | |  | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| **Руководитель** | | | | | |  | | | | | Дата | « » | | 20 |  | г. | |
|  | | | | | | подпись | | | | |  |  | |  |  |  | |

**Аннотация**

К курсовому проекту на тему: «Многопоточная Linux программная модель транспортных перевозок дилижансами по круговому маршруту».

Пояснительная записка к курсовому проекту содержит 31 с., 11 рис.

Составлена укрупнённая схема алгоритма для проекта модели транспортных перевозок дилижансами по круговому маршруту. Детализированы и разработаны схемы алгоритмов для отдельных функций и подзадач.

Ключевые слова: С, дилижанс, пассажир, город, остановка, нить, мьютекс.

Оглавление

[**1.** **Введение** 7](#_Toc41101254)

[**2.** **Описание глобальные переменных** 8](#_Toc41101255)

[**3.** **Описание программы** 9](#_Toc41101256)

[**4.** **Описание работы мьютексов** 10](#_Toc41101257)

[**5.** **Разработка алгоритмов** 11](#_Toc41101258)

[**4.1 Разработка схемы алгоритма функции «main»** 11](#_Toc41101259)

[**4.2 Разработка схемы алгоритма процедуры «flushMap»** 12](#_Toc41101260)

[**4.3 Разработка схемы алгоритма нити пассажира** 14](#_Toc41101261)

[**4.4 Разработка схемы алгоритма нити дилижанса** 17](#_Toc41101262)

[**4.5 Разработка схемы алгоритма нити отрисовки записей** 18](#_Toc41101263)

[**6.** **Скриншоты выполнения работающей программы** 20](#_Toc41101264)

[**7.** **Листинг программы** 26](#_Toc41101265)

[**8.** **Заключение** 31](#_Toc41101266)

[**9.** **Список использованной литературы** 32](#_Toc41101267)

## **Введение**

Целью выполнения курсового проекта является закрепление теоретических знаний и отработка практических навыков работы при разработке многопоточных программ.

Для реализации данной цели необходимо разработать многопоточную программную модель транспортных перевозок дилижансами по круговому маршруту в ОС Linux.

В современных операционных системах широко используются нити (thread), называемые несколько неточно в русском переводе также потоками. Это понятие возникло как результат развития понятия абстрактного процесса. Оказалось, что иногда целесообразно процесс – как некоторую общность программного выполнения, имеющую прикладную цель, – разбить на части, которые выполнялись бы параллельно, конкурируя за главный ресурс – процессор, но в остальном выполняли бы общую работу. Можно подойти к осознанию понятия процесса с другой стороны. Абстрактные процессы теоретически разделяют на конкурирующие и кооперативные. Конкурирующие процессы, по существу, мешают друг другу, но в совокупности выполняют много работ одновременно. Кооперативные процессы выполняют по частям общую работу совместно. Организационно-техническое объединение (под одной "крышей" обобщенного процесса) аналогов кооперативных процессов и составляет существо объединения нитей в одном процессе.

Данная курсовая работа выполнена на языке программирования C. На сегодняшний день C продолжает оставаться одним из наиболее эффективных языков высокого уровня.

## **Описание глобальные переменных**

int countWalking[4] – люди, гуляющие по городу, после приезда на дилижансе.

int countOut – люди, выходящие из дилижанса.

int countOnStop[4] – люди, стоящие на остановке.

int countNextTown – номер следующего города.

int maxPasInDil = 5 – максимальное количество человек в салоне дилижанса.

int countDilSeat – свободные места в дилижансе.

pthread\_mutex\_t dilSeat – мьютекс для учета числа свободных мест.

pthread\_mutex\_t dilOpen – мьютекс для открытия/закрытия дверей у дилижанса.

pthread\_t pas[50] – переменная, которая используется при создании собственной нити у каждого пассажира.

pthread\_t dil – переменная, которая используется при создании нити у дилижанса.

pthread\_t wr – переменная, которая используется при создании нити для отрисовки записей.

pthread\_mutex\_t dilStop[4] – мьютекс для учета числа людей на остановках.

char\* towns[4] = {"Дижон", "Руан", "Париж", "Лион"} – переменная с названиями городов.

char\* words[8] = {"Дилижанс", "Остановка", "На пути в", "Введите количество пассажиров от 1 до 50: ", "Города", "В салоне", "Выходят", "Чтобы выйти из программы нажмите <Esc> и <Enter>"} – переменная, хранящая в себе сообщения, для вывода в консоль.

typedef enum {inCity, onStop, inDil} FlagPas – состояния/флаги, которые показывают, что делает пассажир.

typedef enum {inTown, onWay} FlagDil – состояния/флаги, которые показывают, что делает дилижанс.

## **Описание программы**

В данной программе изображена дорога, связывающая 4 города Дижон, Руан, Париж и Лион. Точка входа в данной программе – функция main. При запуске программы пользователю отправляется запрос на ввод определенного количества пассажиров, которые будут случайным образом распределены на остановках. Минимальное количество пассажиров – 1, максимальное – 50. После ввода, будет вызвана процедура отрисовки карты – flushMap(). В данной процедуре отрисовываются дорога, города, остановки и таблица, в которой будет отображаться все действия, происходящие с дилижансом и пассажирами. Далее создаются нити дилижанса, пассажиров (у каждого своя), а также нить отрисовки записей, которые заносятся в таблицу. После, функция main входит в цикл, который ждет от пользователя, пока он не нажмет комбинацию клавиш <Esc> и <Enter>, после чего программа завершится.

Во время ожидания нажатия данных клавиш, нити пассажиров дилижансов и отрисовки записей, входят в бесконечный цикл. Начнем с нити дилижанса. Дилижанс появляется случайным образом у одной из остановок, после чего отправляется к следующей. По достижению остановки, дилижанс ждет, пока люди не выйдут из салона, путем разблокировки мьютекса dilOpen. Далее проверяется условие, нахождения свободных мест в салоне и нахождения людей на остановке. При выполнении условия пассажиры заходят и усаживаются на свои места, путем разблокировки мьютексов dilSeat и dilStop. После чего данные мьютексы блокируются и дилижанс отправляется на следующую остановку…

Теперь рассмотрим нить пассажира. Изначально человек распределяется на 1 из 4 остановок случайным образом. После чего пассажир встает на остановку и ждет дилижанс. Когда транспорт приезжает, пассажир пытается сесть в него, если есть свободные места. Если условие выполняется, то человек садится в дилижанс, а мьютексы dilSeat и dilStop разблокируются. Если пассажир едет в дилижансе, то разблокируется мьютекс dilOpen. После чего пассажир случайным образом решает выходить ему на следующей остановке или нет. Если человек вышел на остановке, он некоторое время, которое определяется случайным образом, гуляет по городу, а потом возвращается на остановку. И повторяет предыдущие действия…

## **Описание работы мьютексов**

Мьютексы в программе задаются глобально.

pthread\_mutex\_t dilSeat – мьютекс для учета числа свободных мест.

pthread\_mutex\_t dilOpen – мьютекс для открытия/закрытия дверей у дилижанса.

pthread\_mutex\_t dilStop[4] – мьютекс для учета числа людей на остановках.

Инициализация мьютексов происходит в функции «main», после чего мьютексы будут использоваться в нитях дилижанса и пассажира.

Стоя на остановке, пассажир ожидает, когда сможет заблокировать мьютекс, открывающийся только в момент остановки дилижанса, после чего человек попытается занять место в транспорте. Если пассажир зайдет в дилижанс, то разблокирует все занятые мьютексы. Если человек сядет в дилижанс, то заблокирует мьютекс открытия/закрытия двери, закрыв её. На выходе пассажир разблокирует мьютекс открытия/закрытия двери, чтобы выйти из дилижанса.

Пока дилижанс находится в пути, все мьютексы блокируются. Останавливаясь на остановке дилижанс открывает мьютекс открытия/закрытия дверей для выпуска пассажиров, после чего будет пытаться закрыть двери, если поток не занят, с небольшой задержкой. Далее если есть люди на остановке и свободные места в салоне мьютексы отвечающие за учет людей на остановке и учет свободных мест разблокируются, пассажиры зайдут в дилижанс, сядут на свои места, и данные мьютексы заблокируются.

После нажатия клавиш <Esc> и <Enter> в функции «main» все мьютексы освобождаются.

## **Разработка алгоритмов**

## **4.1 Разработка схемы алгоритма функции «main»**

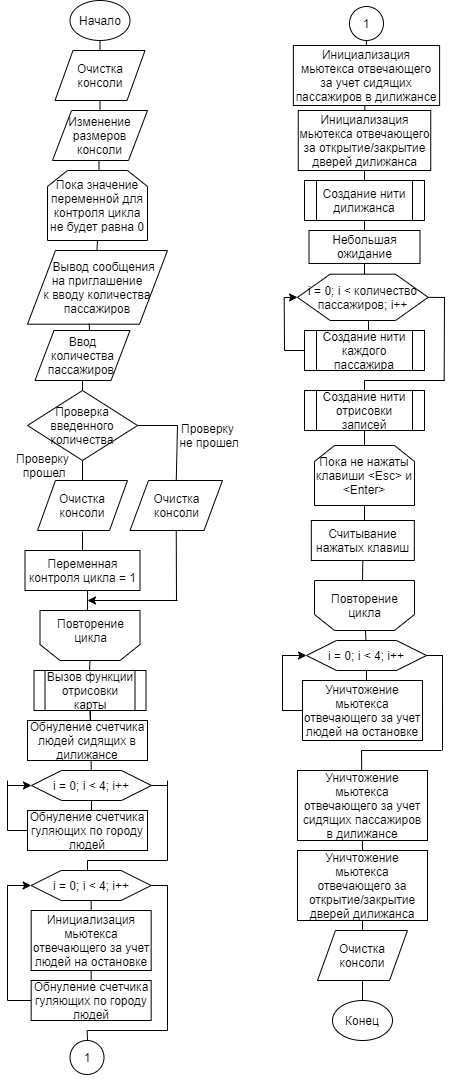


Рисунок 1 – Общая схема алгоритма главной функции

Точка входа в данной программе – функция main, алгоритм данной функции представлен на рис. 1. Первым шагом произойдет очистка консольного окна, после чего изменяться размеры окна 40 на 100. Далее используется цикл, пока пользователь не введет верное количество пассажиров, а именно от 1 до 50 человек. После ввода, будет вызвана процедура отрисовки карты – flushMap(). Далее обнуляются переменные без цикла и через цикл. После чего в цикле инициализируются мьютексы отвечающие за учет людей на остановках и сразу обнуляются. Далее идет инициализация мьютексов отвечающих за учет сидящих пассажиров в дилижансе и за открытие/закрытие дверей у дилижанса. Следующим шагом создаются нити дилижанса, после чего следует небольшая задержка. После, в цикле создаются нити пассажиров (у каждого своя). Далее создается нить отрисовки записей, которые заносятся в таблицу. После, функция main входит в цикл, который ждет от пользователя, пока он не нажмет комбинацию клавиш <Esc> и <Enter>, после чего последует уничтожение мьютексов, очистка консольного окна и завершение программы.

## **4.2 Разработка схемы алгоритма процедуры «flushMap»**

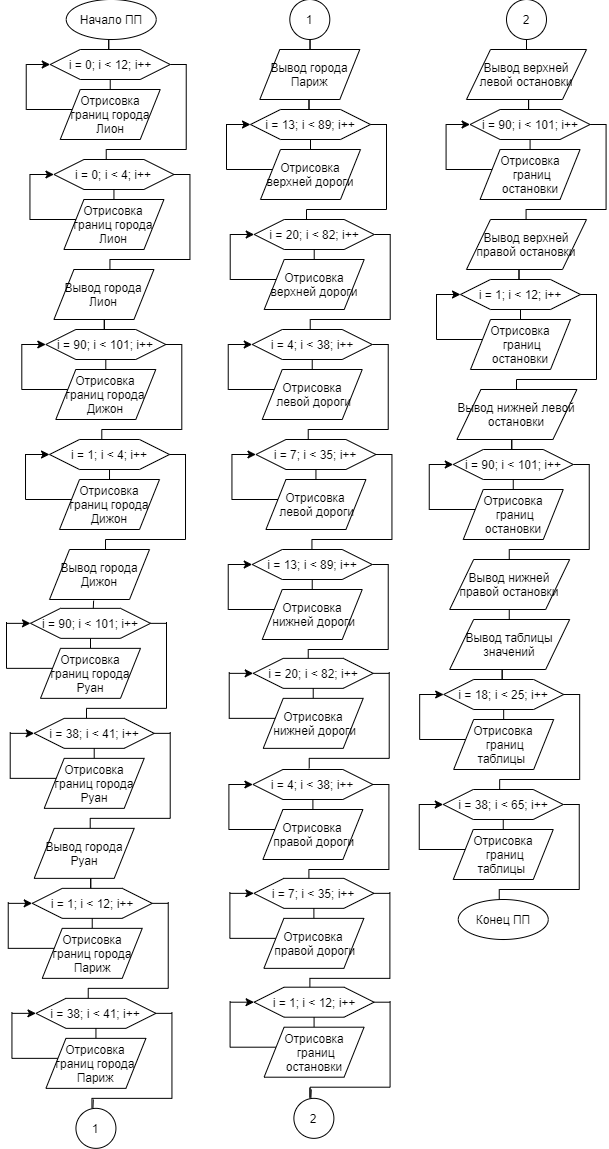


Рисунок 2 – Схема алгоритма функции отрисовки карты

Процедура flushMap, алгоритм данной процедуры представлен на рис. 2. В данной процедуре отрисовывается карта. Используя циклы и функции вывода отрисовывается города Лион, Дижон, Руан и Париж. Следующим шагом отрисовывается дорога, связывающая города. После отрисовки дорог, отрисовываются остановки рядом с городами. Последним этапом данной функции является отрисовка таблицы, которая будет заполнена значениями, отвечающими за действия пассажиров и дилижанса в нити отрисовки записей.

## **4.3 Разработка схемы алгоритма нити пассажира**

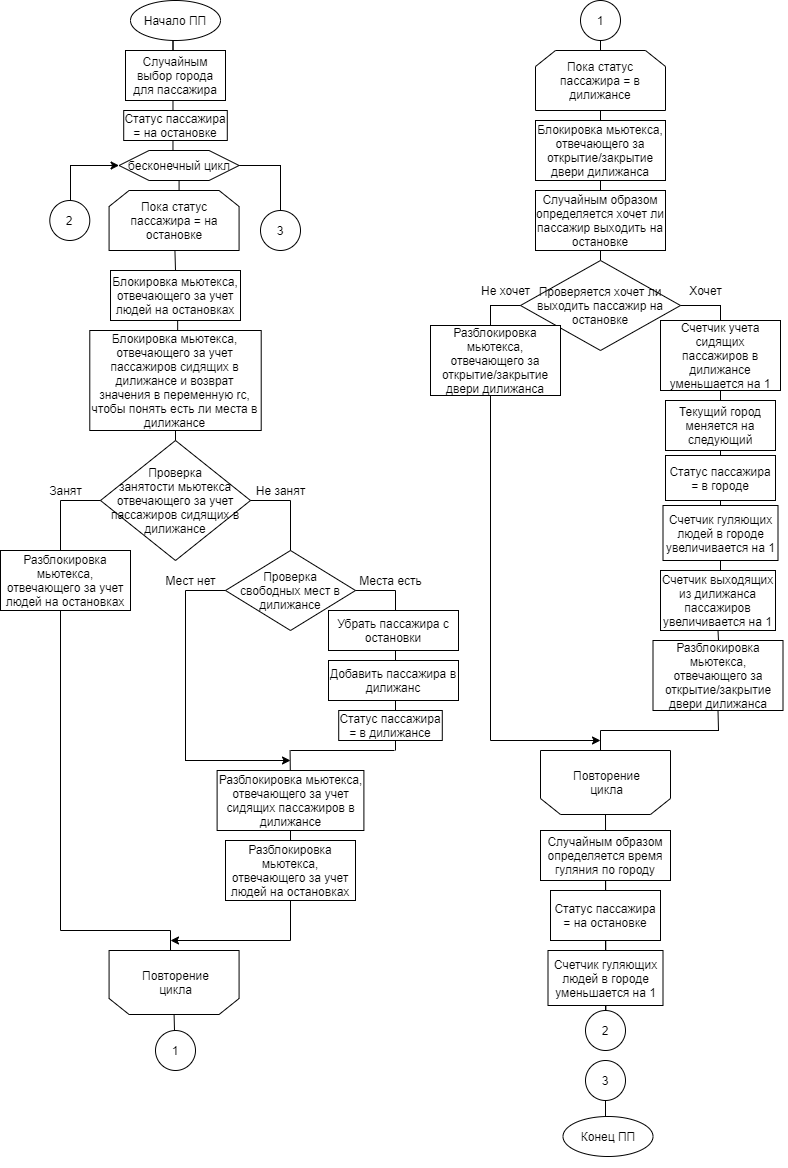


Рисунок 3 – Схема алгоритма нити пассажира

Нить пассажира, алгоритм данной нити представлен на рис. 3. Первым шагом определяется случайным образом в каком, городе будет находиться пассажир, после чего его статус меняется – «на остановке». Далее начинается бесконечный цикл, в котором вызывается сразу следующий цикл с условием, пока статус пассажира = «на остановке», повторять цикл. В данном цикле первым шагом блокируется мьютекс, отвечающий за учет людей на остановке. После чего блокируется мьютекс, отвечающий за учет пассажиров, сидящих в дилижансе, и возвращает значение в переменную – rc, показывающее занят ли поток данного мьютекса или нет. Если данный поток занят, то идет разблокировка мьютекса, отвечающего за учет людей на остановках. А если не занят, то проверяется есть ли свободные места в дилижансе. Если места есть, то пассажир садиться в дилижанс, все переменные, которые вели учет находящихся человек на остановке и в дилижансе изменяются на единицу, а статус данного пассажира меняется на – «в дилижансе». После чего идет разблокировка мьютексов, отвечающих за учет сидящих в дилижансе пассажиров и отвечающих за учет людей на остановке. По завершению данного цикла, начинается новый цикл с условием, пока статус пассажира = «в дилижансе». Первым шагом в данном цикле является блокировка мьютекса, отвечающего за открытие/закрытие дверей дилижанса. После чего случайным образом определяется хочет, ли выходить пассажир на следующей остановке. Если пассажир хочет выйти, то счетчик, отвечающий за учет сидящих пассажиров в дилижансе, уменьшается на единицу, текущий город меняется на следующий, статус пассажира меняется на – «в городе», счетчик гуляющих людей увеличивается на единицу, счетчик выходящих пассажиров из дилижанса увеличивается на единицу. После чего идет разблокировка мьютекса, отвечающего за открытие/закрытие дверей дилижанса. После выхода из цикла случайным образом определяется время хождения по городу. После того, как время выйдет статус пассажира сменится – «на остановке». А счетчик гуляющих людей в городе уменьшится на единицу. Далее цикл повторяется.

## **4.4 Разработка схемы алгоритма нити дилижанса**

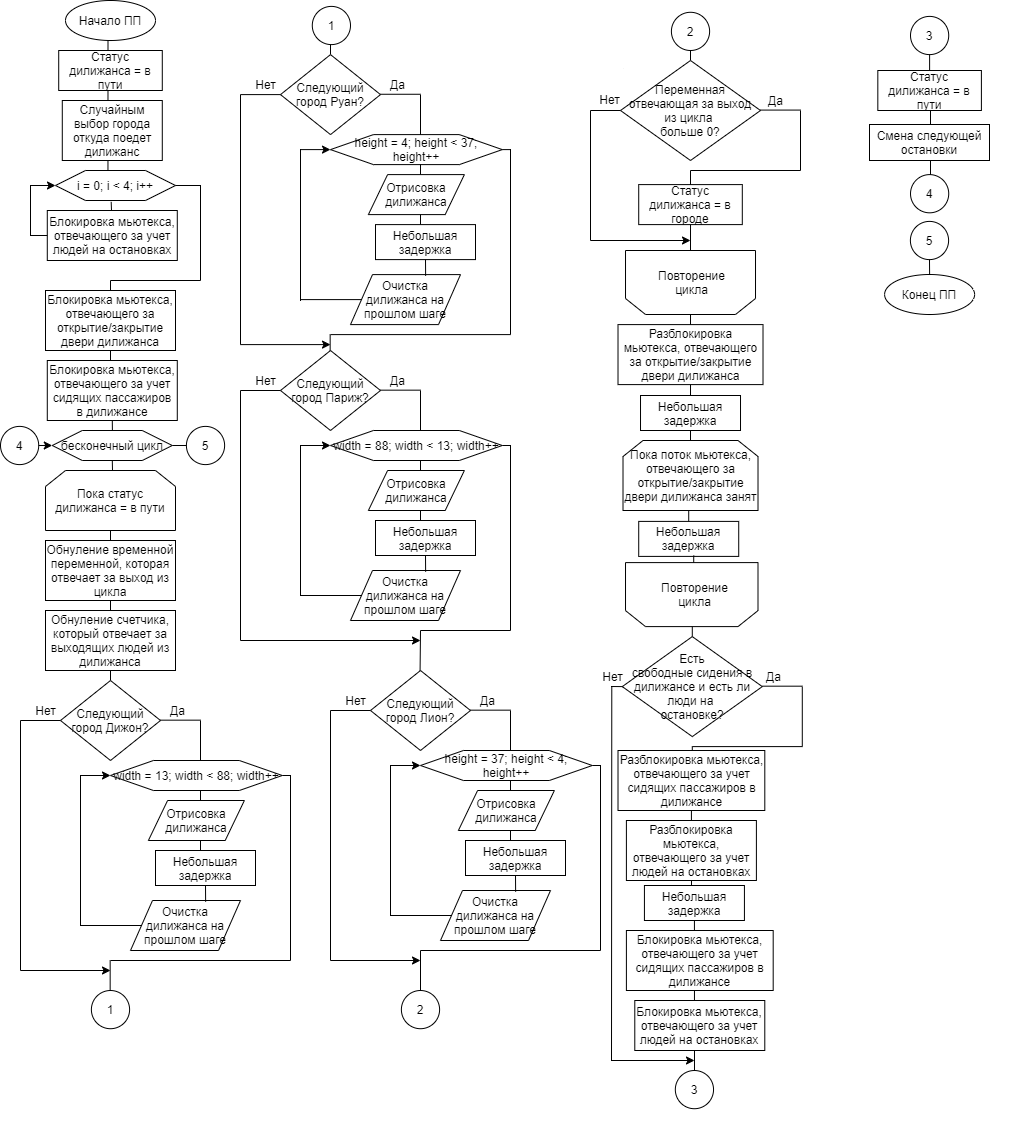


Рисунок 4 – Схема алгоритма нити дилижанса

Нить дилижанса, алгоритм данной нити представлен на рис. 4. Первым шагом статус дилижанса меняется на – «в пути», после чего случайным образом выбирается город, от которого поедет дилижанс. Следующим шагом в цикле блокируются мьютексы отвечающие за учет людей на остановках. После выхода из цикла блокируются мьютексы отвечающий за открытие/закрытие двери дилижанса и отвечающий за учет сидящих пассажиров в дилижансе. Далее дилижанс уходит в бесконечный цикл. После чего сразу входит в другой цикл с условием, пока статус дилижанса = «в пути». Первым шагом в данном цикле является обнуление переменной, отвечающей за выход из цикла, а также обнуление счетчика, который отвечает за выходящих людей из дилижанса. После чего, идет проверка следующего города, и в зависимости от города, выбирается путь отрисовки дилижанса. Если найдена хотя бы одна успешная проверка, то статус дилижанса, меняется на = «в городе». После выхода из цикла идет разблокировка мьютекса, отвечающего за открытие/закрытие двери дилижанса, после чего начинается небольшая задержка. Далее начинается цикл с условием, пока поток мьютекса, отвечающий за открытие/закрытие дверей занят, цикл не закончится. После цикла проверяется условие, есть ли свободные места в дилижансе и есть ли люди на остановке? Если условие выполняется, то следующим шагом разблокируются мьютекс, отвечающий за учет сидящих пассажиров в дилижансе и мьютекс, отвечающий за учет людей на остановках, после чего идет небольшая задержка, чтобы люди зашли в дилижанс. Следующим шагом блокируются мьютекс, отвечающий за учет сидящих пассажиров в дилижансе и мьютекс, отвечающий за учет людей на остановках. После чего меняется статус дилижанса на – «в пути» и идет смена остановки на следующую. Далее цикл повторяется.

## **4.5 Разработка схемы алгоритма нити отрисовки записей**

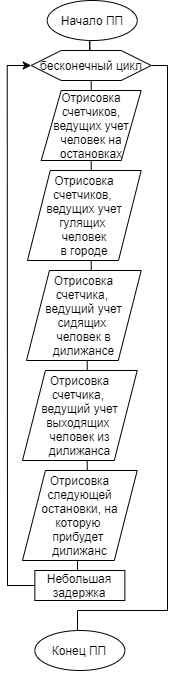


Рисунок 5 – Схема алгоритма нити отрисовки записей

Нить отрисовки записей, алгоритм данной нити представлен на рис. 5. Первым шагом нить сразу входит в бесконечный цикл, после чего идет отрисовка счетчиков, ведущих учет количества человек на остановках, гулящих человек в городе, сидящих человек в дилижансе и выходящих человек из дилижанса, а также отрисовывается следующая остановка, на которую прибудет дилижанс. После чего идет небольшая задержка и цикл повторяется.

## **Скриншоты выполнения работающей программы**

Скриншоты выполнения работы программы изображены на рис. 6 – 11.

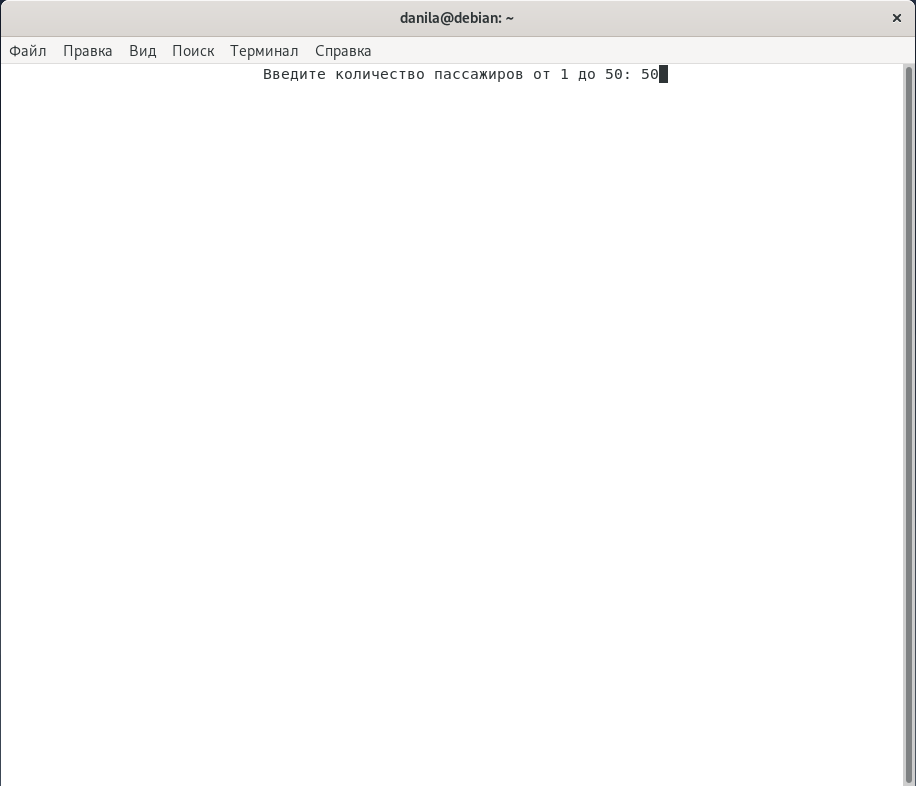


Рисунок 6 – Скриншот ввода количества пассажиров

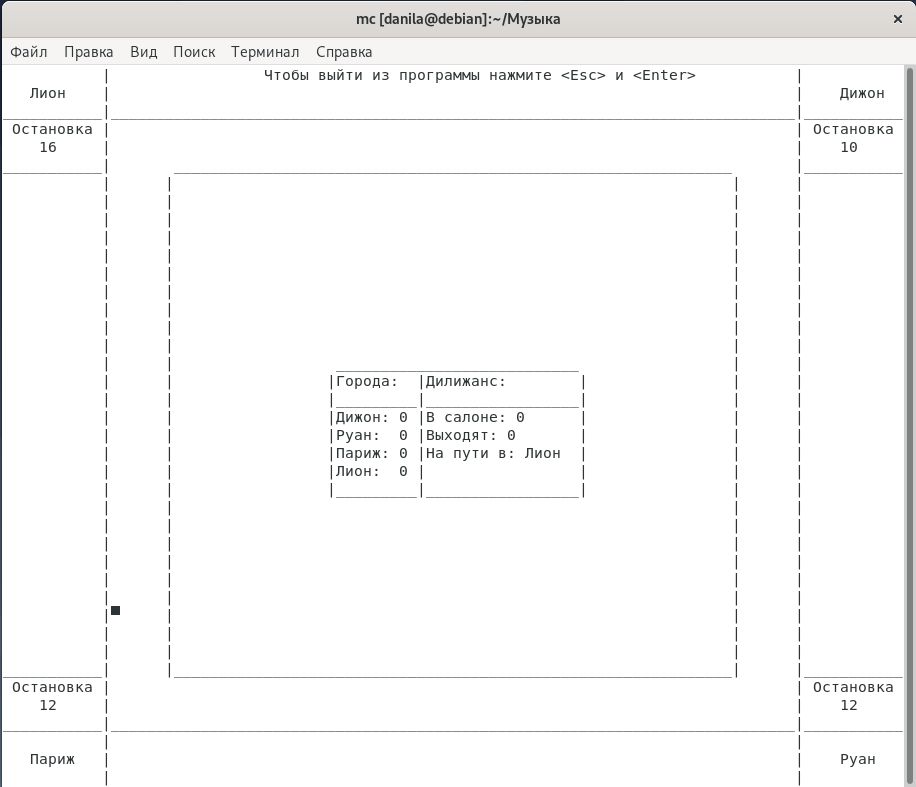


Рисунок 7 – Скриншот появления дилижанса у остановки и распределения людей случайным образом по городам

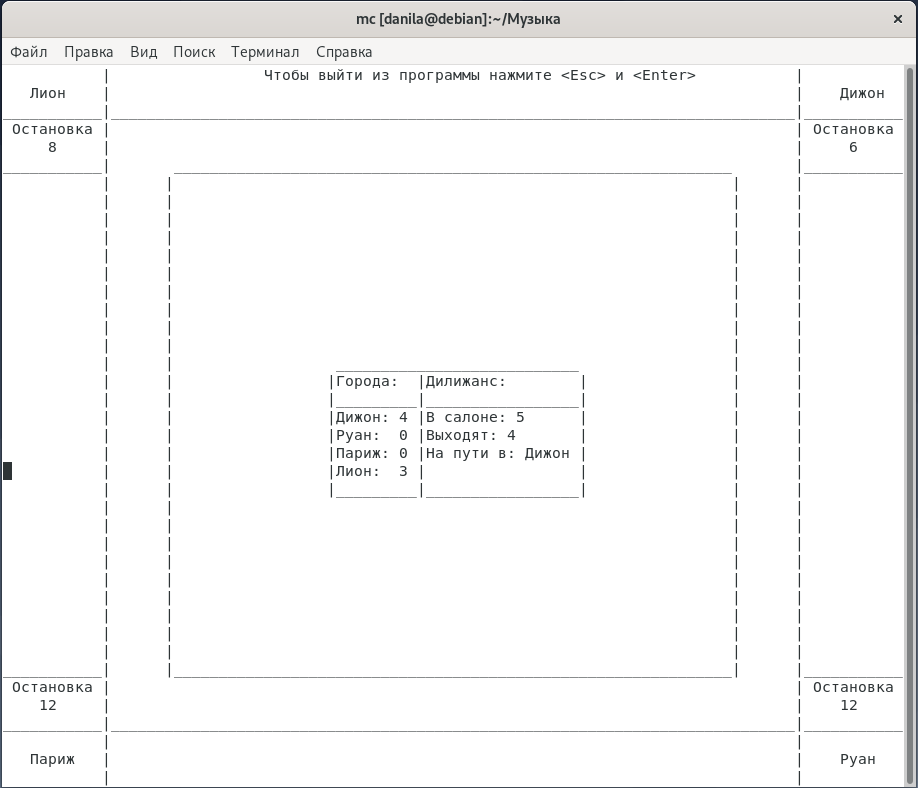


Рисунок 8 – Скриншот выхода из дилижанса пассажиров и прием новых

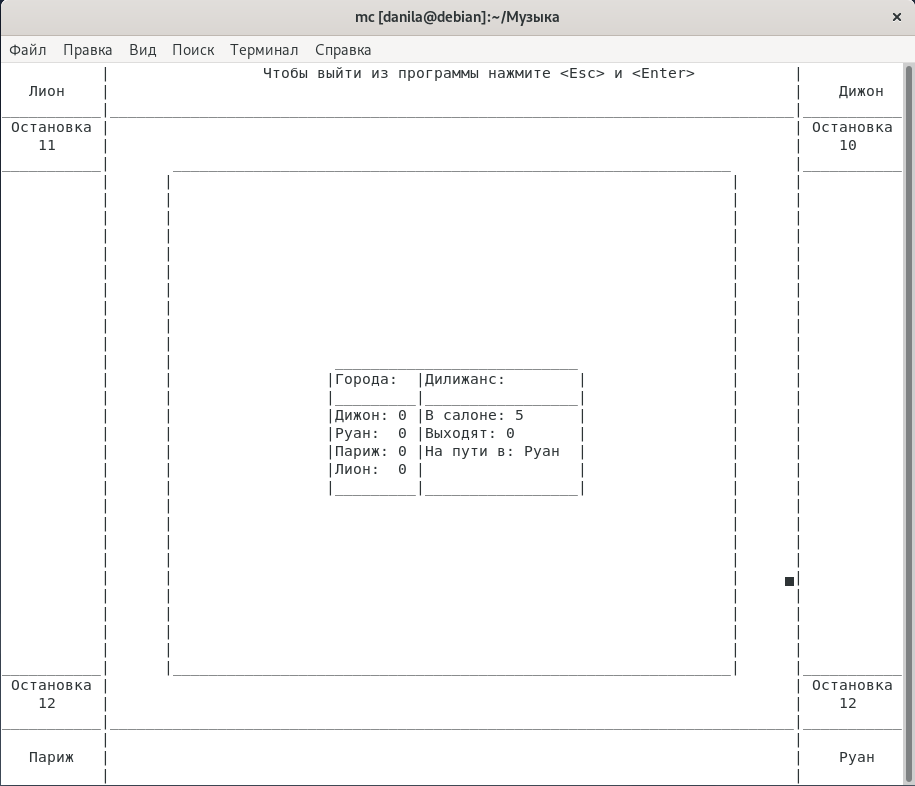


Рисунок 9 – Скриншот дилижанса отправляющегося в Руан

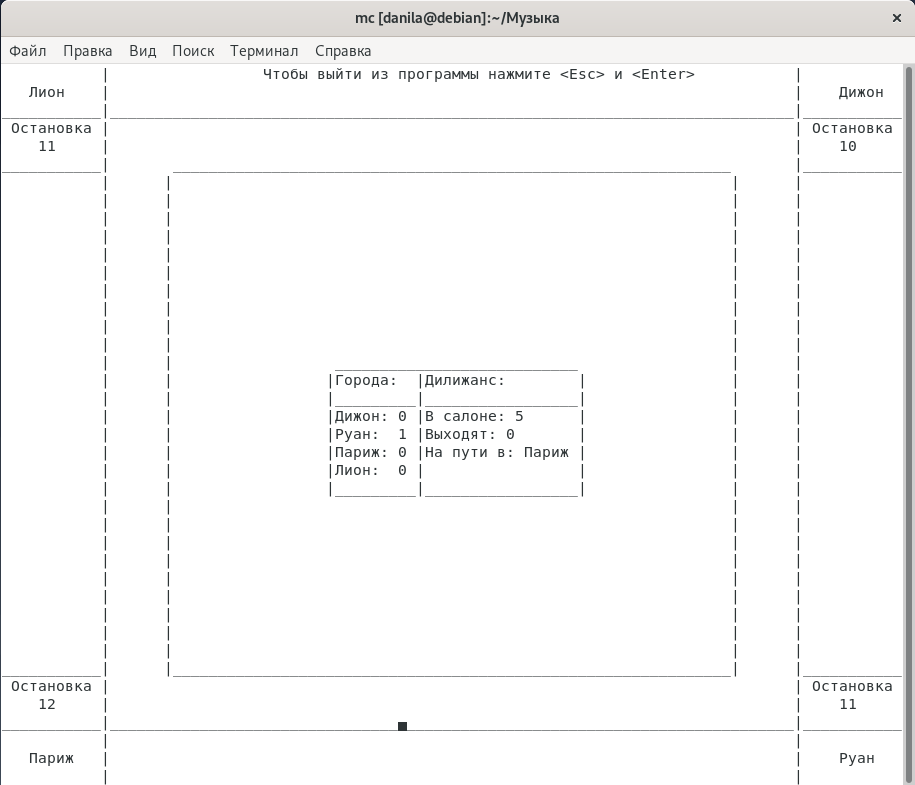


Рисунок 10 – Скриншот дилижанса отправляющегося в Париж

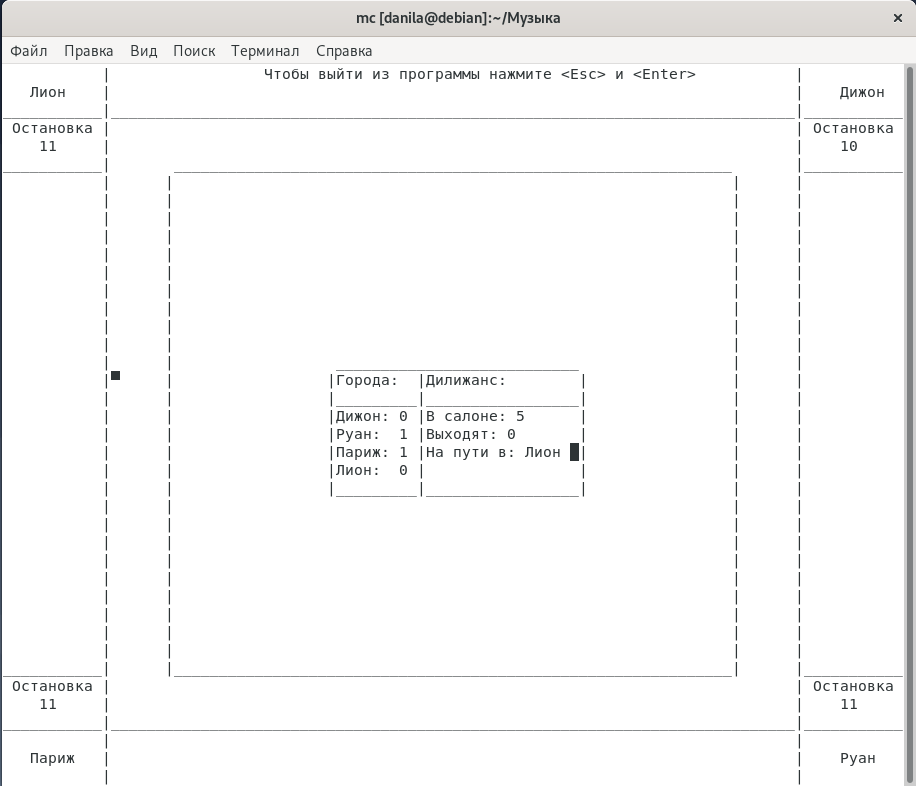


Рисунок 11 – Скриншот дилижанса отправляющегося в Лион

## **Листинг программы**

#include <errno.h>

#include <stdio.h>

#include <time.h>

#include <pthread.h>

#include <stdlib.h>

int countWalking[4];

int countOut;

int countOnStop[4];

int countNextTown;

int maxPasInDil = 5;

int countDilSeat;

pthread\_mutex\_t dilSeat;

pthread\_mutex\_t dilOpen;

pthread\_t pas[50];

pthread\_t dil;

pthread\_t wr;

pthread\_mutex\_t dilStop[4];

char\* towns[4] = {"Дижон", "Руан", "Париж", "Лион"};

char\* words[8] = {"Дилижанс", "Остановка", "На пути в", "Введите количество пассажиров от 1 до 50: ", "Города", "В салоне", "Выходят", "Чтобы выйти из программы нажмите <Esc> и <Enter>"};

typedef enum {inCity, onStop, inDil} FlagPas;

typedef enum {inTown, onWay} FlagDil;

void writer( void \*arg ){ // нить отрисовки записей

for(;;){

printf("\033[5;5H%2d \n", countOnStop[3]);

printf("\033[36;5H%2d \n", countOnStop[2]);

printf("\033[5;94H%2d \n", countOnStop[0]);

printf("\033[36;94H%2d \n", countOnStop[1]);

printf("\033[20;38H%s:%2d \n", towns[0], countWalking[0]);

printf("\033[21;38H%s: %2d \n", towns[1], countWalking[1]);

printf("\033[22;38H%s:%2d \n", towns[2], countWalking[2]);

printf("\033[23;38H%s: %2d \n", towns[3], countWalking[3]);

printf("\033[20;48H%s: %d\n", words[5], countDilSeat);

printf("\033[21;48H%s: %d\n", words[6], countOut);

printf("\033[22;48H%s: %s \n", words[2], towns[countNextTown]);

usleep(10);

}

}

void dilijans(void \*arg) { // нить дилижанса

int i, rc, width, height;

FlagDil flag = onWay;

countNextTown = (4.0 \* (rand()/(RAND\_MAX + 1.0)));

for(i = 0; i < 4; i++) { pthread\_mutex\_lock( &dilStop[i] ); }

rc = pthread\_mutex\_lock(&dilOpen);

rc = pthread\_mutex\_lock(&dilSeat);

for(;;){

while(flag == onWay) {

int temp = 0;

countOut = 0;

if (countNextTown == 0) {

for (width = 13; width <= 88; width++) {

printf("\033[4;%dH%s\n", width, "▀");

usleep(45000);

printf("\033[4;%dH%s\n", width, " ");

temp = 1;

}

}

if (countNextTown == 1) {

for (height = 4; height <= 37; height++) {

printf("\033[%d;88H%s\n", height, "▄");

usleep(90000);

printf("\033[%d;88H%s\n", height, " ");

temp = 1;

}

}

if (countNextTown == 2) {

for (width = 88; width >= 13; width--) {

printf("\033[37;%dH%s\n", width, "▄");

usleep(45000);

printf("\033[37;%dH%s\n", width, "\_");

temp = 1;

}

}

if (countNextTown == 3) {

for (height = 37; height >= 4; height--) {

printf("\033[%d;13H%s\n",height,"▀");

usleep(90000);

if(height != 37) { printf("\033[%d;13H%s\n", height, " ");

} else { printf("\033[%d;13H%s\n", height, "\_"); }

temp = 1;

}

}

if(temp > 0) { flag = inTown; }

}

rc = pthread\_mutex\_unlock(&dilOpen);

usleep(50);

while(pthread\_mutex\_trylock(&dilOpen) == EBUSY){

usleep(50);

}

if((countDilSeat < maxPasInDil)&&(countOnStop[countNextTown] > 0)) {

rc = pthread\_mutex\_unlock(&dilSeat);

rc = pthread\_mutex\_unlock(&dilStop[countNextTown]);

sleep(1);

rc = pthread\_mutex\_lock(&dilSeat);

rc = pthread\_mutex\_lock(&dilStop[countNextTown]);

}

flag = onWay;

countNextTown += 1;

countNextTown = (countNextTown > 3) ? 0 : countNextTown;

countNextTown = (countNextTown < 0) ? 3 : countNextTown;

}

}

void passenger(void \*arg) { // нить пассажира

int rc;

int curTown = (4.0 \* (rand()/(RAND\_MAX + 1.0)));

int myId = (int)arg;

FlagPas flag = onStop;

for(;;) {

countOnStop[curTown]++;

while(flag == onStop) {

rc = pthread\_mutex\_lock(&dilStop[curTown]);

rc = pthread\_mutex\_trylock(&dilSeat);

if(rc == EBUSY) {

rc = pthread\_mutex\_unlock(&dilStop[curTown]);

} else {

if(countDilSeat < maxPasInDil){

countOnStop[curTown]--;

countDilSeat++;

flag = inDil;

}

rc = pthread\_mutex\_unlock(&dilSeat);

rc = pthread\_mutex\_unlock(&dilStop[curTown]);

}

}

while(flag >= inDil) {

rc = pthread\_mutex\_lock( &dilOpen);

float out = 10.0 \* ((rand()/(RAND\_MAX + 1.0)));

if(out > 5.0) {

countDilSeat--;

curTown = countNextTown;

flag = inCity;

countWalking[curTown]++;

countOut++;

rc = pthread\_mutex\_unlock(&dilOpen);

} else {

rc = pthread\_mutex\_unlock(&dilOpen);

sleep(1);

}

}

sleep((int)(10.0 \* (rand()/(RAND\_MAX + 1.0))));

flag = onStop;

countWalking[curTown]--;

}

}

void flushMap() { // процедура отрисовки карты

int i;

for(i = 1; i < 12; i++) { printf("\033[3;%dH\_", i); }

for(i = 1; i < 4; i++) { printf("\033[%d;12H|", i); }

printf("\033[2;4H%s", towns[3]); //Лион

for(i = 90; i < 101; i++) { printf("\033[3;%dH\_", i); }

for(i = 1; i < 4; i++) { printf("\033[%d;89H|", i); }

printf("\033[2;94H%s", towns[0]); //Дижон

for(i = 90; i < 101; i++) { printf("\033[37;%dH\_", i); }

for(i = 38; i < 41; i++) { printf("\033[%d;89H|", i); }

printf("\033[39;94H%s", towns[1]); //Руан

for(i = 1; i < 12; i++) { printf("\033[37;%dH\_", i); }

for(i = 38; i < 41; i++) { printf("\033[%d;12H|", i); }

printf("\033[39;4H%s", towns[2]); //Париж

for(i = 13; i < 89; i++) { printf("\033[3;%dH\_", i); }

for(i = 20; i < 82; i++) { printf("\033[6;%dH\_", i); } // верхняя дорога

for(i = 4; i < 38; i++) { printf("\033[%d;12H|", i); }

for(i = 7; i < 35; i++) { printf("\033[%d;19H|", i); } // левая дорога

for(i = 13; i < 89; i++) { printf("\033[37;%dH\_", i); }

for(i = 20; i < 82; i++) { printf("\033[34;%dH\_", i); } // нижняя дорога

for(i = 4; i < 38; i++) { printf("\033[%d;89H|", i); }

for(i = 7; i < 35; i++) { printf("\033[%d;82H|", i); } // правая дорога

for (i = 1; i < 12; i++) { printf("\033[6;%dH\_", i); }

printf("\033[4;2H%s", words[1]); // верхняя левая остановка

for (i = 90; i < 101; i++) { printf("\033[6;%dH\_", i); }

printf("\033[4;91H%s", words[1]); // верхняя правая остановка

for (i = 1; i < 12; i++) { printf("\033[34;%dH\_", i); }

printf("\033[35;2H%s", words[1]); // нижняя левая остановка

for (i = 90; i < 101; i++) { printf("\033[34;%dH\_", i); }

printf("\033[35;91H%s", words[1]); // нижняя правая остановка

printf("\033[18;38H%s: \n", words[4]); // таблица значений

printf("\033[18;48H%s: \n", words[0]);

for (i = 18; i < 25; i++) { printf("\033[%d;37H|", i); }

for (i = 18; i < 25; i++) { printf("\033[%d;65H|", i); }

for (i = 38; i < 65; i++) { printf("\033[17;%dH\_", i); }

for (i = 38; i < 65; i++) { printf("\033[19;%dH\_", i); }

for (i = 38; i < 65; i++) { printf("\033[24;%dH\_", i); }

for (i = 18; i < 25; i++) { printf("\033[%d;47H|", i); }

printf("\033[1;30H%s", words[7]);

}

int main(int argc, char\*\* argv) {

printf("\033c");

printf("\e[8;40;100;t");

int rc, i, j, maxPas, check = 0, temp = 0;

while(check == 0) {

printf("\033[1;30H%s", words[3]);

scanf("%d", &maxPas);

if((maxPas > 50)||(maxPas < 1)) {

printf("\033c");

} else {

printf("\033c");

check = 1;

}

}

flushMap();

countDilSeat = 0;

for(i = 0; i < 4; i++) { countWalking[i] = 0; }

for(i = 0; i < 4; i++) {

rc = pthread\_mutex\_init(&dilStop[i], NULL);

countOnStop[i] = 0;

}

rc = pthread\_mutex\_init(&dilSeat, NULL);

rc = pthread\_mutex\_init(&dilOpen, NULL);

rc = pthread\_create(&dil, NULL, (void\*)dilijans, NULL);

usleep(50);

for(i=0; i < maxPas; i++) { rc = pthread\_create(&pas[i], NULL, (void\*)passenger, (void\*)i); }

rc = pthread\_create(&wr, NULL, (void\*)writer, NULL);

while(temp != 27){

temp = getchar();

}

for(i = 0; i < 4; i++) {

pthread\_mutex\_destroy(&dilStop[i]);

}

pthread\_mutex\_destroy(&dilSeat);

pthread\_mutex\_destroy(&dilOpen);

printf("\033c");

return 0;

}

## **Заключение**

Была разработана программа, реализующая многопоточную программную модель транспортных перевозок дилижансами по круговому маршруту в ОС Linux. Программные модули были написаны на языке C. В процессе разработки программ были использованы мьютексы, массивы, нити и другие стандартные операторы.

## **Список использованной литературы**

1 Флоренсов А.Н. Операционные системы для программиста: Учеб. пособие. Омск: Изд-во ОмГТУ, 2005. 240 с.