# Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

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

КУРСОВОЙ ПРОЕКТ (РАБОТА)

|  |  |
| --- | --- |
| по дисциплине | Операционные системы |
|  |  |
| на тему | Многопоточная Linux графическая модель имитации работы маршрутного такси |

Пояснительная записка

|  |  |
| --- | --- |
| Шифр проекта (работы) | 020-КП-09.03.01-10-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Студента (ки) | | | | Зайцев Владимир Александрович | | | | | | | | | | | | |  |
|  |  |  |  | | | | фамилия, имя, отчество полностью | | | | | | | | | | | | |  |
|  |  |  | Курс | 2 |  | | | Группа | | | | | ИВТ-172 | | | |  | | |  |
|  |  |  |  |  | |  | | | |  |  | | | | |  | |  | | |
|  |  |  | Направление (специальность) | | | | | | | | |  | | | | | | |  | |
|  |  |  | ***09.03.01 (правильный номер)*** | | | | | | | | | ***Информатика и вычислительная техника*** | | | | | | |  | |
|  |  |  | код, наименование | | | | | | | | | | | | | | | |  | |
|  |  |  | Руководитель | | | | доцент, к.т.н | | | | | | | | | | | | |  |
|  |  |  |  | | | | ученая степень, звание | | | | | | | | | | | | |  |
|  |  |  | Флоренсов Александр Николаевич | | | | | | | | | | | | | | | |  | |
|  |  |  | 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 2019

Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

**ОТЗЫВ**

**на курсовой проект (работу)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Факультет (институт) | | | | | | | | Факультет информационных технологий и компьютерных систем | | | | | | | | | | | | | | |  | |
|  | | | | | | | |  | | | | | | | | | | | | | | |  | |
| Кафедра | | | Информатики и вычислительной техники | | | | | | | | | | | | | | | | | |  | | | |
|  | | |  | | | | | | | | | | | | | | | | | |  | | | |
| Дисциплина | | | | Операционные системы | | | | | | | | | | | | | | | |  | | | | |
|  | | | |  | | | | | | | | | | | | | | | |  | | | | |
| Тема | Многопоточная Linux графическая модель имитации работы маршрутного такси | | | | | | | | | | | | | | | | | | | | | | |  |
|  | |  | | | | | | | | | | | | | | | | | |  | | | | |
| Студент (ка) | | | | Зайцев Владимир Александрович | | | | | | | | | | | | | | | |  | | | | |
|  | | | | фамилия, имя, отчество полностью | | | | | | | | | | | | | | | |  | | | | |
| Курс | 2 | | | | Группа | | | | ИВТ-172 | |  | | | | | | | | | | |  | | |
|  | |  | | | |  | | | |  | |  | | | | | | | | | | |  | |
| Руководитель | | | | | Флоренсов Александр Николаевич | | | | | | | | | | | | | | | | | |  | |
|  | | | | | ученая степень, звание, ФИО | | | | | | | | | | | | | | | | | |  | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
| **Содержание отзыва** | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | | | | | | | | | |  | | | | |
| **Рейтинговые баллы за выполнение и подготовку к защите курсового проекта (работы)** | | | | | | | | | | | | | | | |  | | | | | | | | |
| **Заключение о допуске к защите** | | | | | | | | | | | | |  | | | | | | | | | | | |
| **Руководитель** | | | | | | |  | | | | | | | Дата |  | | 20 |  | г. | | | |  | |

Реферат

Пояснительная записка по курсовому проекту 25 с., 4 ч., 7 рис., 3 источ, 1 прил.

C, LINUX, XLIB, GCC, МНОГОПОТОЧНОЕ ПРИЛОЖЕНИЕ.

Объектом исследования является алгоритм взаимодействия нескольких потоков в операционной системе Linux при работе в графическом окне.

Цель работы – разработка многопоточной графической имитации работы маршрутного такси.

В ходе работы реализован алгоритм взаимодействия нескольких потоков при работе с графическим окном в операционной системе Linux.

В результате была получена программа, которая отображает имитацию работы маршрутного такси и его пассажиров.

Содержание

[Введение 4](#_Toc512456690)

[1 Введение в проблематику разработки многопоточных приложений 5](#_Toc512456691)

[2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения 6](#_Toc512456692)

[3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения](#_Toc512456693) 7

[4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур](#_Toc512456694) 8

[Заключение](#_Toc512456695) 14

[Список использованных источников](#_Toc512456696) 15

[Приложение](#_Toc512456697) 16

Введение

Курсовой проект по дисциплине «Операционные системы», 2 курс. В проекте использовался язык программирования C.

Задача:

Проект состоит из четырех разделов:

* Введение в проблематику разработки многопоточных приложений
* Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения
* Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения
* Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

1 Введение в проблематику разработки многопоточных приложений

При разработке многопоточных приложений возникает ряд задач, которые даже принципиально не могли возникнуть при разработке однопоточных приложений.

Большая проблема, в общем-то, всего одна - одновременный доступ к некоторым ресурсам, которые используются ими в приложении совместно. Решается эта проблема путём введения специального вида объектов, которые управляют взаимодействием потоков.

Если не обеспечить синхронизацию, то вполне можно получить ситуацию, когда один поток удаляет элемент из списка, а второй в то же самое время пытается что-нибудь в этот же самый элемент записать, или же два потока будут записывать информацию в один элемент массива одновременно. Избегают тиких ситуаций путём синхронизации - фактически, перевода приложения на каком-то этапе его выполнения из многопоточного режима в однопоточный, то есть один поток будет делать то, что ему нужно, а остальные станут ждать, пока он закончит. Первый вариант синхронизации - это мьютексы. Как только какой-либо поток добирается в рамках своего выполнения до области действия мьютекса, тот сразу устанавливается в закрытое состояние, после чего остальным потокам вход в зону действия этого мьютекса становится запрещён. Если поток выходит из мьютекса, то, соответственно, меняется и состояние этого объекта синхронизации, которое становится неотмеченным, то есть, по-русски говоря, открытым для всех остальных потоков.

Второй вариант синхронизации - это семафоры. В общем-то, можно сказать, что семафор - это расширенный мьютекс, или же, наоборот, что мьютекс - это частный вид семафора. Семафор аналогичным образом ограничивает вхождение потоков в какую-либо область программного кода, но при этом делает это таким образом, что количество потоков, которым в неё можно войти, не ограничивается одним. Для этого семафор использует внутренний счётчик потоков, который увеличивается на единицу при вхождении потока в область действия семафора и уменьшается на ту же самую единицу тогда, когда поток и семафор расстаются. При достижении максимального числа потоков все остальные, желающие попасть туда, будут ожидать, пока вакантное место не освободится.

В случае разработки многопоточных приложений для UNIX систем, необходимо подключение бибилиотеки pthread.h а так же указание -lpthread при компиляции, чтобы компилятор подсоединил необходимый функционал.

2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения

Основной частью разрабатываемого приложения явлется алгоритм управляющий пассажирами пользующимися маршрутным такси. Данный функционал был выделелен в отдельную процедуру peopleThreadFunction. Сама эта функция так же подразделена на меньшие подзадачи для каждого из действий, которое может совершать симулируемый человек.

Следующая часть разрабатываемого приложения — это алгоритм маршрутного такси. Так же как и предыдущая часть программы, эта задача была разделена на более мелкие подзадачи, только в этот раз их меньше, т. к. алгоритм работы такси несколько проще.

Часть программы реализующая отображение данной модели в графическом окне выполнена в функции showContent. Данная функция реализует как графическое отожбражение модели в виде условных схем и значков, так и отладочное отображение моели в текстовом виде, оформленное в виде таблицы.

Часть программы, связывающаяя все функции описанные выше, будет заключена в основной процедуре main в основном потоке. В данной процедуре будет осуществляться инициализация окна X сервера, его настройка, создание средств для отображения графики в данном окне, к примеру таких, как палитры цветов, и сами цвета. В этой же процедуре осуществляется считывание пользовательского ввода, после чего создается необходимое количество пассажиров, и потокв для них. Следует отметить, что по факту алгоритм работы такси так же находится в процедуре main внутри бесконечного цикла. К такому решению привел тот факт, что в соответствии с заданем такси в программе всего одно, следовательно рациональнее всего будет предоставить основной поток именно ему.

3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения

targer\_framerate — переменная необходимая для настройки скорсости и плавности отображения модели, а так же для решения проблемы с мерцанием отображения.

Dspl — ссылка на дисплей X сервера

hwnd — ссылка на окно X сервера

event - пременная предназначеная для хранения событий окна

key\_sym — переменная необходимая для преобразования нажимаемых пользователем клавишь клавиатуры в символы типа char

gc — графический контекст, необходимый для отображения графики в окне X сервера

hmtx — мьютекс для синхронизации доступа пассажиров к маршрутному такси

l\_green — blue — цвета используемые при построении графического отображения симуляции

stop\_size — размер остановки

coord — структура необходимая для хранения координаты

stop — структура для хранения информации о каждой остановке

stops\_list — массив всех остановок

people — структура необходимая для хранения информации о человеке

taxi — структра для хранения информации о маршрутном такси

taxi1 — единственное маршрутное такси участвующее в модели

people\_list — массив содержащий информацию о всех людях участвующих в симуляции

people\_count — количество людей которое вводит пользователь

4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

В общем случае архитектура данного приложения совмещает в себе две сущности. Модель симулируемой ситуации, и отображение данной ситуации. Фактически данную архитектуру можно отнести к паттерну «модель, отображение, контроллер»(MVC), т. к. отображение отделено от самой модели, то ничего не мешает при желании реализовать в данном приложении иной способ отображения или логирования симулируемого процесса. Это возможно благодаря тому, что данные о симулируемых обьектах строго систематезированны с помощью использования стрктур, а также благодря тому, то все эти данные храянтся в специально отведенных для них массивах, что занчительно упрощает доступ к ним.

Так как была поставлена задача симулировать достаточно динамичный процесс, было принято решение, что самым наглядным будет реализовать анимированное гарфическое представление данного процесса. Т.к. анимация — по сути это многократное изменение различных параметров объектов с течением времени, то самым удобным инструментом здесь представляются циклы. Изменяя различные параметры объектов,(к примеру координаты) несколько раз в секунду можно добиться иллюзии движения. В данном проекте такие изменения происходят приблизительно 60 раз в секунду. Делать это чащще не имеет смысла, т. к. эффект не будте заметен изза аппаратных ограничений оборудования (в основном монитора).

Каждый поток запущенный основным потоком модифицирует данные находящиеся в массиве people\_list. Это могут быть не только координаты людей, но и различные флаги, предназначенные для определения действий, которые должен соврешать человек. Не смотря на то, что какалось бы все эти потоки будут обращаться к одному и томуже массиву, и что такие обращения могут происходить одновременно, нет никакой необходимости в синхронизации этих потоков между собой по доступу к этому массиву. Это обьясняется тем, что каждый поток модифицирует строго только одну ячейку массива и не имеет никакого доступа к остальрным. Данное решение значительно упрощает реализацию поставленной задачи.

Тем не менее средства синхронизации потоков было необходимо применить для осуществления совместного доступа этих потоков к одной перменной, входящей в состав структуры описывающей маршрутное такси. Эта перменная предназначена для хранения информации о текущем количестве пассажиров находящихся в такси. Без использования данного решеня возникала проблема, когда в такси могло находится больше человек, чем оно могло перевозить.

Как было написано выше, графическое отображене полностью отделено от самой модели. В функции showContent с помощью цикла итерирующегося по всем элементам массива с людьми, происходит получение данных об этих людях, их координат, и флагов состояния. На основе полученных данных строится изображение состоящее из множества геометрических фигур расположенных в тех координатах которые были получены ранее, и окрашенных в разные цвета в соответствии с состояниями этих объектов. Так же для наглядности данная модель отображается не только в графическом предствалении но и с помощью форматированной таблицы с текстом в правой части экрана. При желании можно написав дополнительный модль для данной программы осуществить вывод этих данных в консоль или в текстовый файл.

Схема алгоритма главной процедуры представлена на рисунке 1.

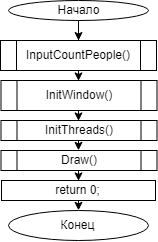


Рисунок 1 – схема алгоритма основной процедуры

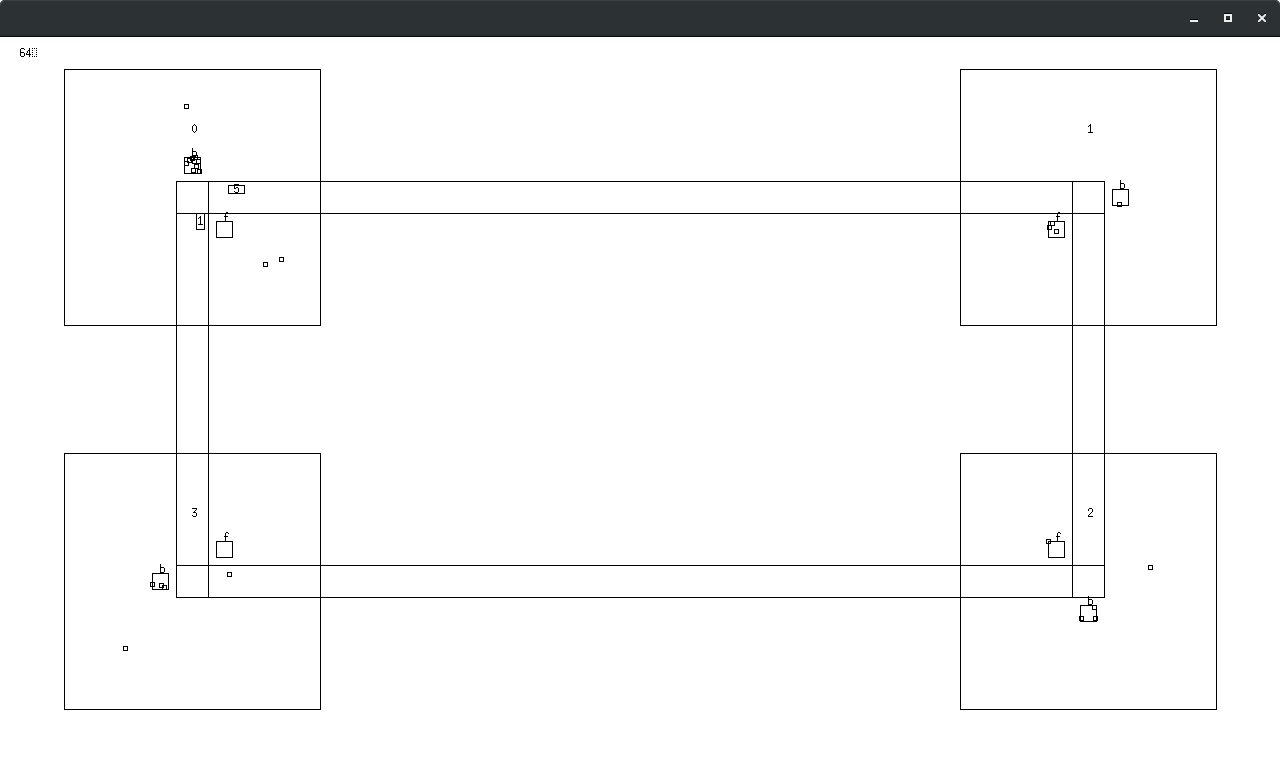
Скриншот работы программы представлен на рисунках 2-6.

Рисунок 2 – пример работы программы

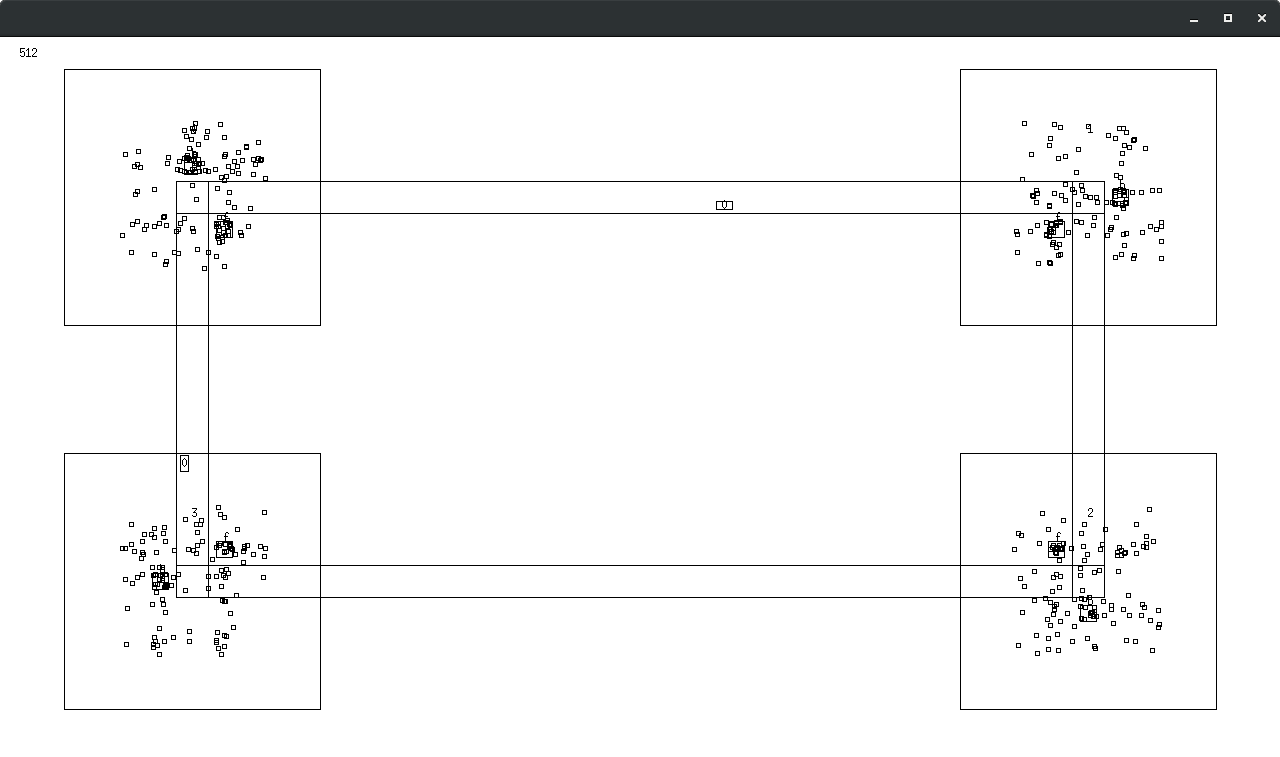


Рисунок 3 – пример работы программы

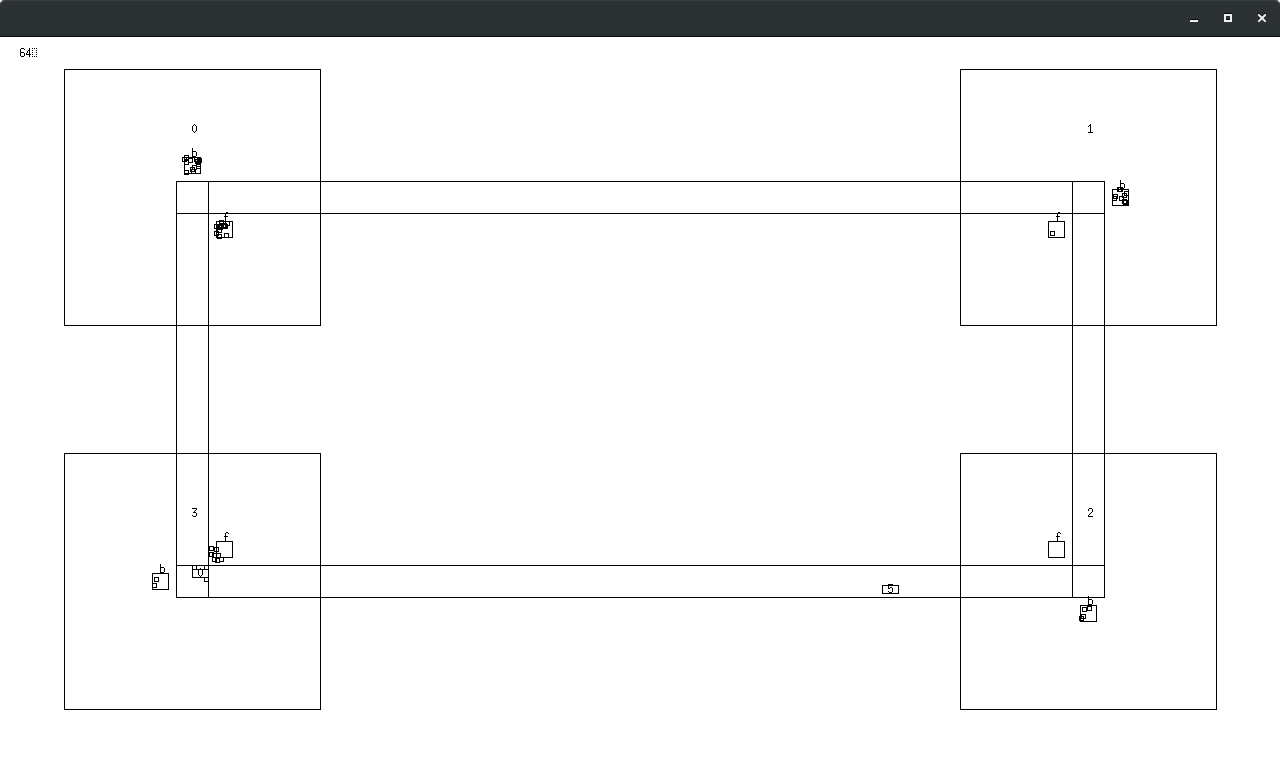


Рисунок 4 – пример работы программы

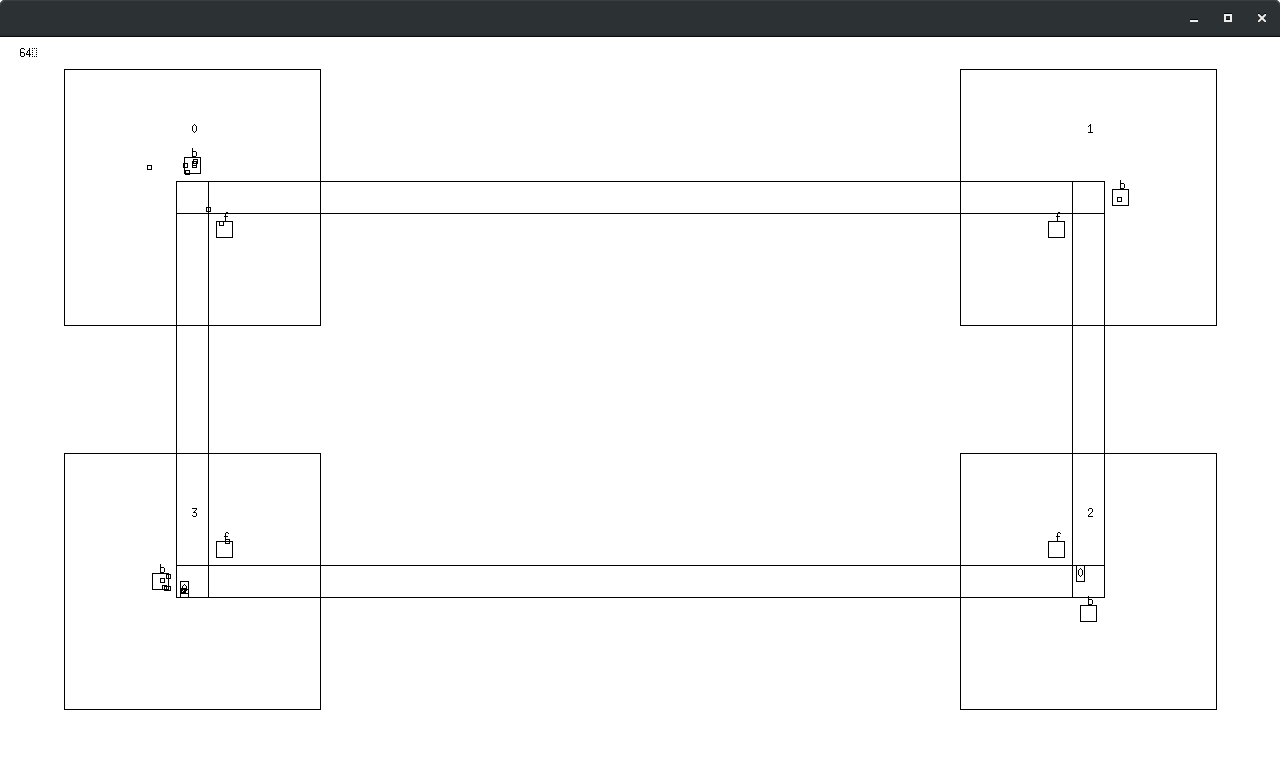
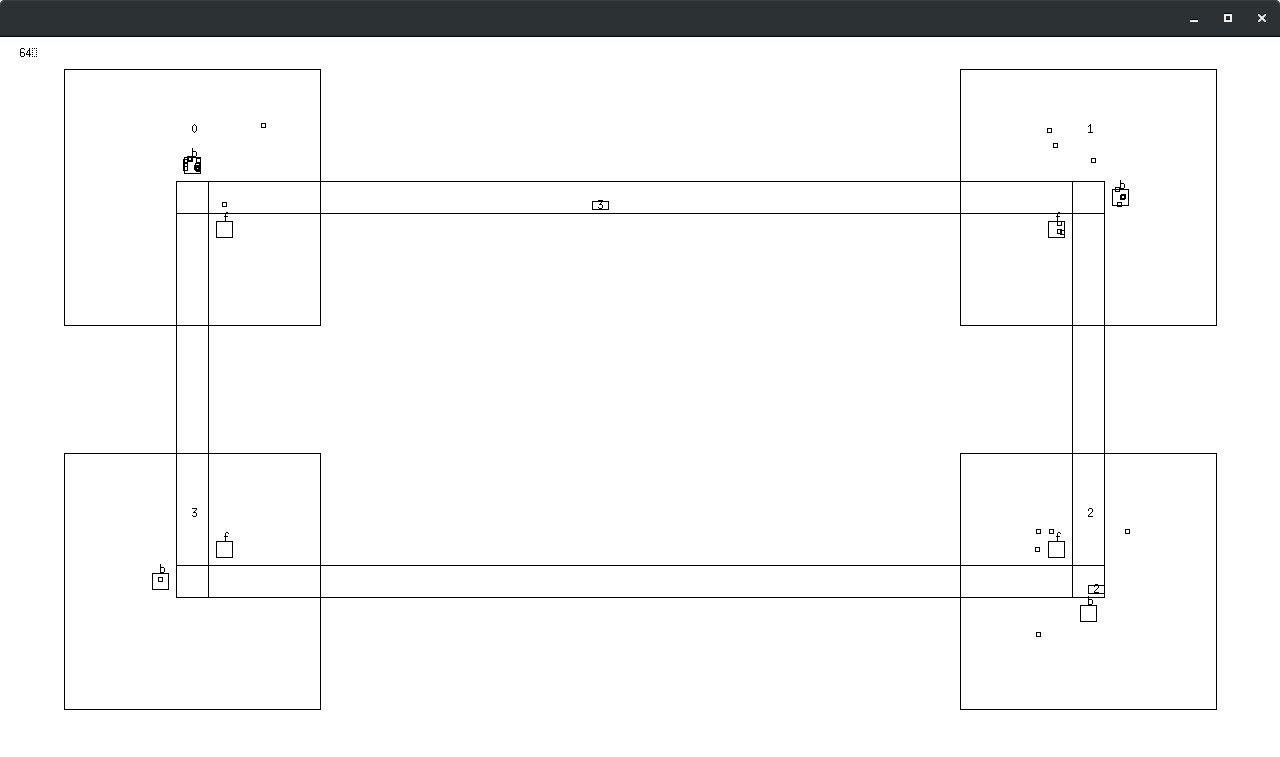
Рисунок 5 – пример работы программы

Рисунок 6 – пример работы программы

Скриншот файла для трансляции программы представлен на рисунке 7.

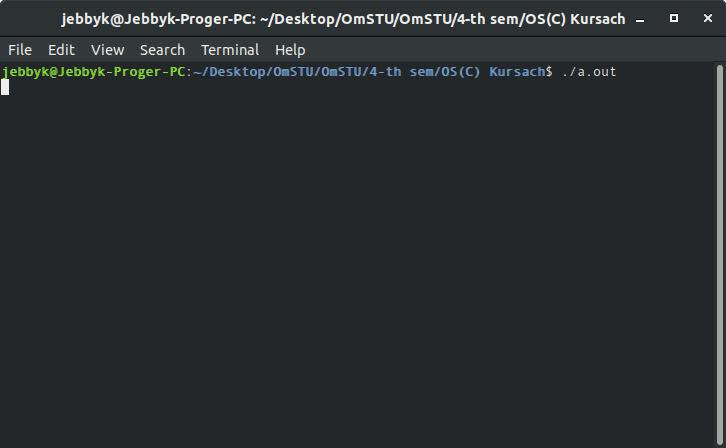


Рисунок 7 – cкриншот файла для трансляции программы

Заключение

Список использованных источников

1. Флоренсов, А.Н. Операционные системы для программиста. Омск. ОмГТУ, 2005.
2. Гордеев, А.В. Операционные системы / А.В. Гордеев. – 2-е изд. – СПб.: Питер, 2007
3. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

Приложение

**Исходный код программы**

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#include <X11/Xlib.h>//библиотека для графики

#include <math.h>//библиотека для математических функций

unsigned short target\_framerate = 60;//плавность и скорость работы

Display \*dspl;//ссылка на дисплей для рисования

int screen;

Window hwnd;//ссылка на окно в котором происходит рисование

XEvent event;//события окна

KeySym key\_sym;//переменная для ковертирования клавиш в цифрыы

GC gc;//ссылк на обьект для рисования

pthread\_mutex\_t hmtx;//мьютекс для блокировки доступа к такси

XColor l\_green, d\_green, white, gray, d\_gray, yelow, red, blue;//используемые цвета

unsigned int stop\_size = 16;//размер остановки

typedef struct{//структура для координат

float x;

float y;

}coord;

typedef struct {//структура для описсания остановки

coord position;//координаты остановки

coord stop\_frwd;//координаты остановочного места в одном направлении

coord stop\_bkwd;//координаты остановочного места в обратном направлении

}stop;

stop stops\_list[4];

void SetStops()//расположение всех остановок и остановочных мест для каждой остановки

{

stops\_list[0].position.x = 64;//координаты самих остановок

stops\_list[0].position.y = 200;

stops\_list[1].position.x = 300;

stops\_list[1].position.y = 128;

stops\_list[2].position.x = 600;

stops\_list[2].position.y = 300;

stops\_list[3].position.x = 756;

stops\_list[3].position.y = 150;

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

{

stops\_list[i].stop\_frwd.x = stops\_list[i].position.x;//координаты остановочных мест для каждой остановки (для движения в двух направлениях)

stops\_list[i].stop\_frwd.y = stops\_list[i].position.y - 16;

stops\_list[i].stop\_bkwd.x = stops\_list[i].position.x;

stops\_list[i].stop\_bkwd.y = stops\_list[i].position.y + 16;

}

}

typedef struct{//структура описывающая персонажей

coord position;//текущая позиция персонажа

coord target\_position;//кординаты точки, куда нужно идти человеку

short drive;//флаг говорящий о том что человвек едет

short sleep;//спит

short walk\_stop;//идет к остановке

short walk\_taxi;//идет к такси

short walk\_out;//идет от остановки

short wait;//ждет

short direction;//направление в котором хочет ехать человек

short curent\_stop;//текущая остановка, на которой находится человек

float speed;//текущая скорость перемещения человека

}people;

typedef struct{//структура описывающая такси

coord position;//текущая позиция

coord target\_position;//точка в которую едет такси

float speed;//его скорость

short drive;//флаг, который указывает, едет ли сейчас такси

short wait;//ждет на остановке

short people\_amount;//количество лудй внутри

short curent\_stop;//текущая остановка, на которой находится такси

short direction;//направление движения такси

}taxi;

taxi taxi1;//само такси

people people\_list[1024];//массив содержащий множество структур со всеми людьми

short people\_count;//переменная хранящая количесттво людей котрое ввел пользователь

void peopleThreadFunction(void \*\_i)//функция с алгоритмом поведения персонажей

{

int i = \_i;//сохраняемый переданный функции аргумент, это будет номер элемента массива с людьми, к оторому у функции будте доступ

people\_list[i].curent\_stop = rand()%4;//задаем рандомную остановку при старрте

people\_list[i].direction = rand()%2;//задаем ранодомное направление

if(people\_list[i].curent\_stop == 0) people\_list[i].direction = 0;//если остановка - крайняя левая, то направление - вперед

if(people\_list[i].curent\_stop == 3) people\_list[i].direction = 1;//если остановка - крайняя правая, то направление - обратное

if(people\_list[i].direction == 0) people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_frwd;//указываем точку движения персонажа соответствующие

if(people\_list[i].direction == 1) people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_bkwd;//координаты остановочных мест

people\_list[i].target\_position.x += -8 + rand()%16;//добавляем рандомный разброс, чтоб толпились на остановке

people\_list[i].target\_position.y += -8 + rand()%16;

people\_list[i].position = stops\_list[people\_list[i].curent\_stop].position;//указываем начальные координаты самих персонажей

people\_list[i].position.x += -64 + rand()%128;

people\_list[i].position.y += -64 + rand()%128;

people\_list[i].walk\_stop = 1;//ставим флаг, что персоонаж движется к остановке

while(1)//бесконечный цикл для непрерыной обработки персонажей

{

if(people\_list[i].walk\_stop)//елси текущий флаг - движение к остановке

{

coord dir\_vector = {people\_list[i].target\_position.x-people\_list[i].position.x, people\_list[i].target\_position.y-people\_list[i].position.y};//рассчитываем вектор для движеня персонажа

float l = sqrt(pow(dir\_vector.x,2)+pow(dir\_vector.y, 2));//рассчитываем длину этого вектора

coord normal\_v = dir\_vector;//нормализованный вектор, который должен иметь то же направление, но длинну 1

normal\_v.x /= l;//для этого делим его на длину первого вектора

normal\_v.y /= l;

if(people\_list[i].speed < 0.5 && people\_list[i].speed < l/20)//если скорость персонажа меньше 0.5 и в принципе достаточно млаа

{

people\_list[i].speed += 0.01;//то постеменно ее увеличиваем

}

if(people\_list[i].speed > l/20)//если скорость слишком велика по отношению к оставшемуся расстоянию до цели,

{

people\_list[i].speed = l/20;//то снижаем ее

}

people\_list[i].position.x += normal\_v.x \* people\_list[i].speed;//изменяем координаты персонажа на основании его скорости и нормализованного вектора длвижения

people\_list[i].position.y += normal\_v.y \* people\_list[i].speed;

if(people\_list[i].speed < 0.001)//если персонаж остановился

{

people\_list[i].walk\_stop = 0;//сбрасываем флаг, что он движется к остановке

people\_list[i].wait = 1;//устанавливаем флаг ожидания транспорта

}

}

if(people\_list[i].wait > 0)//если активен флаг ожидания

{

//смотрим, не переполнено ли такси, остановилось ли оно, и находится ли такси на той же остановке, что и персонаж

if(people\_list[i].direction == taxi1.direction && people\_list[i].curent\_stop == taxi1.curent\_stop && taxi1.people\_amount < 4)

{

people\_list[i].walk\_taxi = 1;//елси это так, то устанавливаем флаг, что персонаж движется к такси

people\_list[i].wait = 0;//сбрасываем флаг ожидания транспорта

people\_list[i].target\_position = taxi1.position;//утсанавливаем точку перемещения персонжа - координаты такси

}

}

if(people\_list[i].walk\_taxi == 1)//если активен флаг передвижения к такси

{

if(taxi1.people\_amount < 4 && taxi1.wait > 0)//если такси все еще ждет и там есть свободные места

{

coord dir\_vector = {people\_list[i].target\_position.x-people\_list[i].position.x, people\_list[i].target\_position.y-people\_list[i].position.y};//активируем алгоритм перемещения

float l = sqrt(pow(dir\_vector.x,2)+pow(dir\_vector.y, 2));

coord normal\_v = dir\_vector;

normal\_v.x /= l;

normal\_v.y /= l;

if(people\_list[i].speed < 0.5 && people\_list[i].speed < l/20)

{

people\_list[i].speed += 0.01;

}

if(people\_list[i].speed > l/20)

{

people\_list[i].speed = l/20;

}

people\_list[i].position.x += normal\_v.x \* people\_list[i].speed;

people\_list[i].position.y += normal\_v.y \* people\_list[i].speed;

if(people\_list[i].speed < 0.01)//и если человек почти дошел

{

pthread\_mutex\_lock(&hmtx);//захватываем мьютексб на доступ к такси

people\_list[i].drive = 1;//устанавливаем флаг езды

taxi1.people\_amount++;//увеличиваем у такси информацию о количестве пассажирова

people\_list[i].walk\_taxi = 0;//сбрасываем флаг движения к такси

pthread\_mutex\_unlock(&hmtx);//отпускаем мьютекс

}

}else{//елси такси уже поехало, или там уже закончились места

people\_list[i].walk\_stop = 1;//идем обратно на туже остановку

people\_list[i].walk\_taxi = 0;//сбарсываем флаг движения к такси

if(people\_list[i].direction == 0)//в зависимости от текущего направления, в котром решил ехать персонаж

{

people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_frwd;//выбираем координаты нужного остановочного места как цлеевые/

people\_list[i].target\_position.x += -8 + rand()%16;//добавляем немного рандома для создания толпы

people\_list[i].target\_position.y += -8 + rand()%16;

}

if(people\_list[i].direction == 1)//в зависимости от текущего направления, в котром решил ехать персонаж

{

people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_bkwd;//выбираем координаты нужного остановочного места как цлеевые/

people\_list[i].target\_position.x += -8 + rand()%16;//рандом для толпы

people\_list[i].target\_position.y += -8 + rand()%16;

}

}

}

if(people\_list[i].drive == 1)//есди установлен флаг движени в такси

{

people\_list[i].position = taxi1.position;//приравнивем координаты пассажира к координатам такси

if(taxi1.wait && people\_list[i].curent\_stop != taxi1.curent\_stop)//еслли такси остановилось, и остановка не равна предыдущей остановке, на которой былл человек

{

people\_list[i].curent\_stop = taxi1.curent\_stop;//обновляем информацию о текущей остановке персонажа

int r = rand()%2;//делаем рандом который вернет 1 или 0

if(r == 1)//елс 1 то

{

pthread\_mutex\_lock(&hmtx);//захватываем мьютекс

people\_list[i].drive = 0;//сбрасываем флаг езда на такси

people\_list[i].walk\_out = 1;//устанавливаем флаг движения от остановки

taxi1.people\_amount--;//уменьшаем информациию о количестве пассажиров в такси

people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].position;//указываем целевую точку движения - координаты текущей остановки

people\_list[i].target\_position.x += -64 + rand()%128;//и добавляем большой разброс в координтаах

people\_list[i].target\_position.y += -64 + rand()%128;

pthread\_mutex\_unlock(&hmtx);//отпускаем мьютекс

}

}

}

if(people\_list[i].walk\_out == 1)//если установлен флаг движения в направлении от становки

{

coord dir\_vector = {people\_list[i].target\_position.x-people\_list[i].position.x, people\_list[i].target\_position.y-people\_list[i].position.y};//запускается алгорим перемещения

float l = sqrt(pow(dir\_vector.x,2)+pow(dir\_vector.y, 2));

coord normal\_v = dir\_vector;

normal\_v.x /= l;

normal\_v.y /= l;

if(people\_list[i].speed < 0.5 && people\_list[i].speed < l/20)

{

people\_list[i].speed += 0.01;

}

if(people\_list[i].speed > l/20)

{

people\_list[i].speed = l/20;

}

people\_list[i].position.x += normal\_v.x \* people\_list[i].speed;

people\_list[i].position.y += normal\_v.y \* people\_list[i].speed;

if(people\_list[i].speed < 0.01)//елси дошел до цлеи

{

people\_list[i].walk\_out = 0;//сбрасыаем флаг движения от остановки

people\_list[i].sleep = 1;//устанавливаем флаг сна

usleep(100000\*(rand()%100 + 100));//засыпаем на неопределенное количество времени

people\_list[i].sleep = 0;//после сна сбрасываем флаг сна

people\_list[i].walk\_stop = 1;//устанавливаем флаг движения к остановке снова

people\_list[i].direction = rand()%2;//делаем рандомное направление в котором хочет ехть человек

if(people\_list[i].curent\_stop == 0) people\_list[i].direction = 0;//елси текущая остановка - крайняя то точно указываем направление в котором вообще можно ехать

if(people\_list[i].curent\_stop == 3) people\_list[i].direction = 1;

if(people\_list[i].direction == 0)//в зависимости от выбранного направления

{

people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_frwd;//выбираем нужное остановочное место

people\_list[i].target\_position.x += -8 + rand()%16;//и добавляем рандом для создания толпы

people\_list[i].target\_position.y += -8 + rand()%16;

}

if(people\_list[i].direction == 1)

{

people\_list[i].target\_position = stops\_list[people\_list[i].curent\_stop].stop\_bkwd;

people\_list[i].target\_position.x += -8 + rand()%16;

people\_list[i].target\_position.y += -8 + rand()%16;

}

}

}

usleep(1000000/target\_framerate);//отправляем поток в сон на 1 шестидесятую секунды.

}

}

void showContent()//фнукция для отображения всего происходящего через окно

{

XClearWindow(dspl, hwnd);//сначал очищщаем экран

XSetForeground(dspl, gc, yelow.pixel);//устанавливаем цвет рисования - желтый

XFillRectangle(dspl, hwnd, gc, taxi1.position.x-8, taxi1.position.y - 4, 16, 8);//заполняем прямоугльник такси желтым

coord op;//вспомогательная координата предыдущей остановки для рисования дороги

for(int i = 0; i < 4; i++)//для всех четырех остановок

{

XSetForeground(dspl, gc, red.pixel);//устанавливаем красный цвет рисования

coord p1 = stops\_list[i].stop\_bkwd;//берем координты остановчного места для обратного движения

XDrawRectangle(dspl, hwnd, gc, p1.x-stop\_size/2, p1.y-stop\_size/2, stop\_size, stop\_size);//и рисуем эту осттановку

XSetForeground(dspl, gc, l\_green.pixel);//для остновочного место с обычным направлением движения берем зеленую кисть

coord p2 = stops\_list[i].stop\_frwd;//и выбираем координаты нужного остановочного метса

XDrawRectangle(dspl, hwnd, gc, p2.x-stop\_size/2, p2.y-stop\_size/2, stop\_size, stop\_size);

coord p = stops\_list[i].position;//вспомогательная координата текущей остановки

XSetForeground(dspl, gc, d\_gray.pixel);//выбираем серую кистьдля рисования дороги

if(i > 0)

{

XDrawLine(dspl, hwnd, gc, op.x, op.y, p.x, p.y);//рисуем линию от предыдущей остановки к текущй

}

op = p;//координаты текущей остановки после рисованя становятся координатами предыдущей

}

char pc[4] = "";

sprintf(pc, "%d", people\_count);//преобразуем количество людей в строку

for(int i = 0; i < people\_count; i++)//проходим циклом по всему ссписку людей в масиве

{

if(people\_list[i].sleep == 1) XSetForeground(dspl, gc, d\_gray.pixel);//елси человек спит, то рисуем его серым

else XSetForeground(dspl, gc, white.pixel);//иначи рисуем белым

XFillArc(dspl, hwnd, gc, people\_list[i].position.x - 2, people\_list[i].position.y - 2, 4, 4, 0,360\*64);//куржок символизирующий персонажа

char c[2] = "";

sprintf(c, "%d", i);//преобразуем текущий номер элемента массива в букву

XSetForeground(dspl, gc, white.pixel);//устанавливаем цвет рисования - белый

XDrawString(dspl, hwnd, gc, 820, 16 + i \* 18, c, 2);//пишем номер человека в нужных координатах

XSetForeground(dspl, gc, d\_gray.pixel);//устанавливаем цвет рисованя - серый

if(people\_list[i].walk\_stop == 1) {//если человек сейчкс идет к остановке

XSetForeground(dspl, gc, d\_green.pixel);//устанавливаем темно зеленый цвет рисования

XDrawString(dspl, hwnd, gc, 840, 16 + i \* 18, "wk\_st", 5);//выводим текст в таблицу

}

if(people\_list[i].walk\_taxi == 1) {//делаем так же с остальными флагами в таблице, разукрашивая их в разные цвета для наглядности

XSetForeground(dspl, gc, yelow.pixel);

XDrawString(dspl, hwnd, gc, 880, 16 + i \* 18, "wk\_tx", 5);

}

if(people\_list[i].walk\_out == 1) {

XSetForeground(dspl, gc, blue.pixel);

XDrawString(dspl, hwnd, gc, 920, 16 + i \* 18, "wk\_ot", 5);

}

if(people\_list[i].wait == 1) {

XSetForeground(dspl, gc, d\_gray.pixel);

XDrawString(dspl, hwnd, gc, 960, 16 + i \* 18, "wait", 4);

}

if(people\_list[i].drive == 1) {

XSetForeground(dspl, gc, yelow.pixel);

XDrawString(dspl, hwnd, gc, 1000, 16 + i \* 18, "drive", 5);

}

if(people\_list[i].direction == 1)

{

XSetForeground(dspl, gc, red.pixel);

XDrawString(dspl, hwnd, gc, 1060, 16 + i \* 18, "bkwd", 4);

}

if(people\_list[i].direction == 0)

{

XSetForeground(dspl, gc, l\_green.pixel);

XDrawString(dspl, hwnd, gc, 1100, 16 + i \* 18, "frwd", 4);

}

XSetForeground(dspl, gc, d\_green.pixel);

char num[1] = "";

sprintf(num, "%d", people\_list[i].curent\_stop);

XDrawString(dspl, hwnd, gc, 1140 + people\_list[i].curent\_stop \* 12, 16 + i \* 18, num, 1);//отображение номера текущей остановки

}

XFlush(dspl);// отправление графической информации в само окно на вывод

}

void main()

{

dspl = XOpenDisplay(NULL);//создание окна

gc = XDefaultGC(dspl,0);//дескриптор для рисования в графическом окне

if(dspl == 0) {printf("Error XOpenDisplay\n"); exit(1);}//выводим ошибку если не получиолсь соткрыть дисплей

screen = XDefaultScreen(dspl);//сохранаяем инфо о экране

hwnd = XCreateSimpleWindow(dspl, RootWindow(dspl, screen), 100,50,1200,600,3, WhitePixel(dspl,screen), BlackPixel(dspl,screen));//созаем простое графическое окно с размерами 1200 на 600

Colormap screen\_colormap;//созадем новую палитру цветов

screen\_colormap = DefaultColormap(dspl, DefaultScreen(dspl));//копируем то что есть по умолчанию

XAllocNamedColor(dspl, screen\_colormap, "green", &l\_green, &l\_green);//у добавляем туда свои цвета

XAllocNamedColor(dspl, screen\_colormap, "gold", &yelow, &yelow);

XAllocNamedColor(dspl, screen\_colormap, "web gray", &d\_gray, &d\_gray);

XAllocNamedColor(dspl, screen\_colormap, "red", &red, &red);

XAllocNamedColor(dspl, screen\_colormap, "white", &white, &white);

XAllocNamedColor(dspl, screen\_colormap, "dark green", &d\_green, &d\_green);

XAllocNamedColor(dspl, screen\_colormap, "sky blue", &blue, &blue);

XSetForeground(dspl, gc, yelow.pixel);//утсанавливаем желлтый цвет рисования

XSetBackground(dspl, gc, BlackPixel(dspl, screen));//черный цвет фона

if(hwnd == 0) {printf("Error XCreateSimpleWindow\n"); exit(1);}//если произошли ошибки с окном то выводим предупреждение и выходим

XSelectInput(dspl, hwnd, ExposureMask | KeyPressMask );//здес указываем какой ввод с мыши и клавиатуры мы будем считывать во время работы программы

XMapWindow(dspl, hwnd);

int n = 0;//флаг для ввода цифр

char people\_count\_string[4] = "";//строка для хранения вводимого числа

while(n != -1)//пока не сброшен флаг

{

XNextEvent(dspl, &event);//считыавем события от окна Х сревера

if(event.type == KeyPress)//если событие - это нажатие клавиши на клавиатурпе

{

char kb[1];

XLookupString(&event.xkey, kb,1, &key\_sym, 0);//преобразуем его в сивол

char c = kb[0];

int i = c - '0';//потом из символа в цифру

if(i >= 0 && i <=9)//если она в пределах от 0 до 9

{

XDrawString(dspl, hwnd, gc, 10 + n\*10, 10, kb, 1);//выводим на дисплей его

people\_count\_string[n] = kb[0];//записываем в сстроку

n++;// увеличиваем флаг чисел

}else{

n = -1;//елси ввели не знам между 0 и 9 то сбрасываем флаг ввода чисел

sscanf(people\_count\_string, "%d", &people\_count);//и сохраняем количество персонажей

}

}

}

SetStops();//выполняем расположение остановк и остановочных мест в соответствии с координатами

for(int i = 0; i < people\_count; i++)//запускаем цикл на количество итераций равному числу, которое ввел польователь

{

people p;//создаем переменную для человека

pthread\_t thread;//перменную для потока

pthread\_create(&thread, NULL, (void\*)peopleThreadFunction, (void\*)i);//запускаем поток

}

//дальше идет алгоритм для маршрутного такси

taxi1.drive = 1;//усатнавливаем, что при старте программы такси едет

taxi1.direction = 1;//указываем направление (на первой же остановке оно поменяется)

taxi1.curent\_stop = 1;//и сотановка

taxi1.target\_position = stops\_list[0].position;//укаываем начальную координату

while(1)//запускаем безконечный цикл

{

if(taxi1.drive == 1)//в нем проверяем флаг езды, и еси он 1, то

{

coord dir\_vector = {taxi1.target\_position.x-taxi1.position.x, taxi1.target\_position.y-taxi1.position.y};//выполняем перемещение так же как и у пассажиров, только с более высокой скоростью

float l = sqrt(pow(dir\_vector.x,2)+pow(dir\_vector.y, 2));

coord normal\_v = dir\_vector;

normal\_v.x /= l;

normal\_v.y /= l;

if(taxi1.speed < 10 && taxi1.speed < l/20)//сдесь ограничение скорости выставленно в 2 раза выше чем у человеков

{

taxi1.speed += 0.05;

}

if(taxi1.speed > l/20)

{

taxi1.speed = l/20;

}

taxi1.position.x += normal\_v.x \* taxi1.speed;

taxi1.position.y += normal\_v.y \* taxi1.speed;

if(taxi1.speed < 0.001)//елси такси почти полностью осттановилось, то

{

taxi1.drive = 0;//останавливаем его полностью, сбрасывае флаг езды

taxi1.wait = 240;//устанавливая вемя ожиданя

if(taxi1.direction == 0)//дальше проверяем, направление движени

{

taxi1.curent\_stop++;//прибавляем 1 к информации о ткущей остановке такси

if(taxi1.curent\_stop == 3) //еси она оказалась на крайней остановке

{

taxi1.direction = 1;//изменяем направление движения на обратное

}

}else{//тут вспе тоже самое для только для обратного направления движения

taxi1.curent\_stop--;

if(taxi1.curent\_stop == 0)

{

taxi1.direction = 0;

}

}

}

}

if(taxi1.wait > 0)//елси еще не закончислоь время ожидания

{

taxi1.wait--;//то ждем,и немного это времяуменьшаем

if(taxi1.wait == 0) //елси время истекло

{

taxi1.drive = 1;//устанавливаем флаг езды

int c;

if(taxi1.direction == 0)//в зависимости от направления движения

{

c = taxi1.curent\_stop + 1;////выбираем слудующую останвоку

}

if(taxi1.direction == 1)

{

c = taxi1.curent\_stop - 1;

}

taxi1.target\_position = stops\_list[c].position;//передаем ее координаты в целевую точку для движения

}

}

showContent();//выполняем отображене всех людей, такси и остановоок в графическом окне

usleep(1000000/target\_framerate);//отправляем основной поток вв сон на одну шестидесятую секунды

}

getchar(); printf("\033[0m");

}