МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель проекта
_____/Караваева О. В./
(подпись) (Ф.И.О)
«__» _____2020 г.

РАЗРАБОТКА ПРОГРАММНОЙ МОДЕЛИ ПЛАНИРОВЩИКА ПРОЦЕССОВ

Пояснительная записка курсового проекта по дисциплине
«Комплекс знаний бакалавра»
ТПЖА.09.03.01.066 ПЗ

Разработал студент группы ИВТ-42 _____/Рзаев А. Э./

Руководитель старший преподаватель _____/Караваева О. В./

Проект защищен с оценкой «_____» _____
(оценка) (дата)

Члены комиссии _____ / _____/
(подпись) (Ф.И.О)
_____ / _____/
(подпись) (Ф.И.О)
_____ / _____/
(подпись) (Ф.И.О)

Киров 2020

УДК 004.4

Реферат

Рзаев А. Э. Разработка программной модели планировщика процессов: ТПЖА.09.03.01.066 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Караваева О. В. – Киров, 2020. – Гр. ч. 2 л. ф. А2, 2 л. ф. А3; ПЗ 51 с., 13 рис., 3 прил.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ, УПРАВЛЕНИЕ ПРОЦЕССАМИ, ПЛАНИРОВЩИК ПРОЦЕССОВ, ОПЕРАЦИОННЫЕ СИСТЕМЫ, ГЕНЕРАТОР ЗАДАНИЙ, QT, CMAKE, C++, JSON.

Цель курсового проекта – повышение качества обучения за счет использования программного комплекса моделирования работы диспетчера процессов в операционных системах при выполнении лабораторных работ по дисциплине «Операционные системы».

Программное обеспечение, разработанное в рамках данного курсового проекта – лабораторная установка «Модель диспетчера процессов операционной системы».

В ходе выполнения курсового проекта был выполнен анализ предметной области, проектирование и разработка программного обеспечения.

Содержание

Введение

В настоящее время в области образования все больше производится автоматизация контроля знаний учащихся и освоения нового материала.

Одним из способов автоматизации являются специальные программные модели, эмулирующие работу какой-либо системы. С их помощью студенты имеют возможность достаточно подробно изучить ее работу, принципы и особенности. В совокупности с теоретическим материалом это позволяет увеличить степень освоения новых знаний по данной дисциплине и повысить качество обучения в целом.

Такие программные модели достаточно широко используются при выполнении лабораторных работ по дисциплине «Операционные системы». К сожалению, качество некоторых приложений оставляет желать лучшего, что затрудняет изучение нового материала. Поэтому было принято решение выполнить анализ и доработку наиболее проблемной модели – диспетчера процессов операционной системы.

1 Анализ предметной области

На данном этапе работы необходимо провести обзор текущей программной модели диспетчера процессов, выяснить ее недостатки, обосновать актуальность разработки новой модели и сформировать требования к программному обеспечению в виде технического задания.

1.1 Обзор текущей программной модели

Текущая программная модель была разработана в 2002 году студентами Вятского государственного университета А. С. Гордиенко и М. Н. Томчуком; доработана в 2006 году студентами А. Ю. Соколовым и И. А. Брызгаловым.

1.1.1 Подсистема управления процессами

Функциями подсистемы управления процессами являются:

- планирование выполнения процессов и потоков (задач);
- создание и уничтожение процессов;
- обеспечение процессов необходимыми системными ресурсами;
- поддержание взаимодействия между процессами.

При выполнении этих функций подсистема управления процессами взаимодействует с другими подсистемами ОС, ответственными за управление ресурсами:

- подсистема управления памятью;
- подсистема ввода/вывода;
- файловая подсистема.

Планирование выполнения процессов включает в себя решение следующих задач:

– определение момента времени для смены текущего активного потока;

– выбор потока для выполнения из очереди готовых потоков.

В большинстве ОС универсального назначения планирование осуществляется динамически, то есть решения принимаются во время работы системы на основе анализа текущей ситуации. За счет этого динамические планировщики могут гибко приспосабливаться к изменяющейся ситуации.

Статическое планирование может быть использовано в специализированных системах (например, в системах реального времени), в которых весь набор одновременно выполняемых задач определен заранее.

Статический планировщик принимает решение о планировании не во время работы системы, а заранее. Результатом работы такого планировщика является таблица, в которой указывается к какому потоку или процессу, когда и на какое время должен быть предоставлен процессор.

Для выполнения задач планирования процессов ОС должна получить управление при наступлении следующих событий:

– прерывание от таймера (истечение кванта времени);

– запрос на ввод/вывод или на доступ к ресурсу, который сейчас занят. Задача переходит в состояние ожидания;

– освобождение ресурса. Планировщик проверяет, не ожидает ли этот ресурс какая-либо задача. Если да, то эта задача переводится из состояния ожидания в состояние готовности;

– аппаратное прерывание, которое сигнализирует о завершении периферийным устройством операции ввода/вывода, соответствующая задача переводится в очередь готовых и выполняется перепланирование;

– внутреннее прерывание, сигнализирующее об ошибке, которая произошла в результате выполнения активной задачи. Планировщик снимает задачу и выполняет перепланирование;

– запросы приложений и пользователей на создание новой задачи или повышения приоритета существующей задачи.

Для изучения функций планировщика была разработана текущая программная модель. Студентам предлагается выполнить задание, состоящее из последовательности заявок (событий), которые необходимо обработать, выступив в роли планировщика.

1.1.2 Интерфейс пользователя

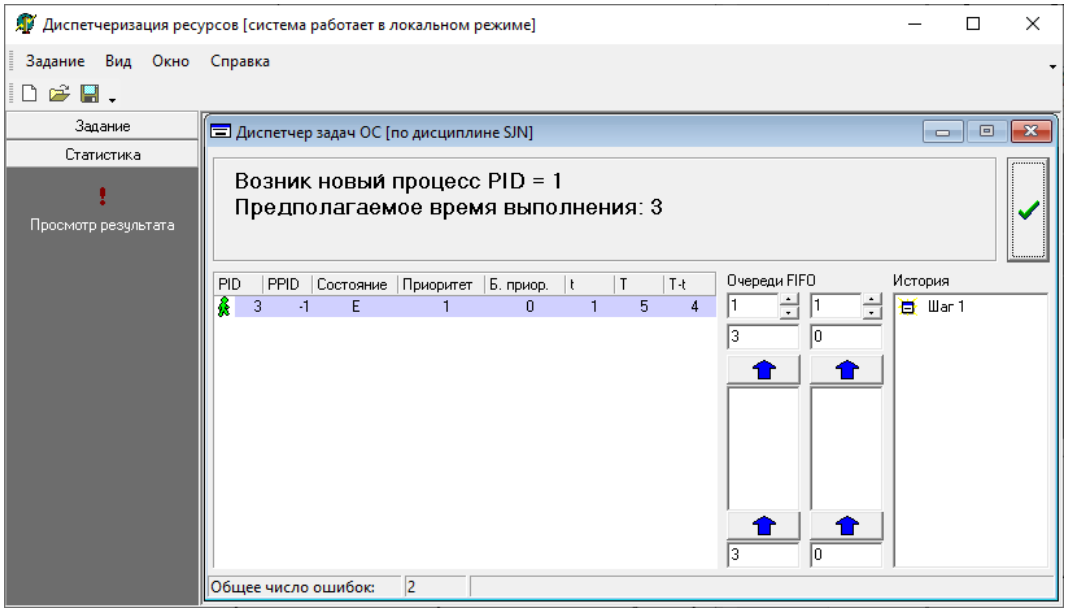


Рисунок 1 – Основное окно программы

Главное окно программы разделено на следующие области:

– заголовок с названием дисциплины;
– описание текущей заявки;
– список процессов;
– блок очередей процессов;
– кнопка «Подтвердить»;
– список обработанных пользователем заявок;
– строка статуса.

В верхней части окна присутствует описание заявки и количество выполненных заявок. Задача пользователя – корректно обработать поступающие заявки.

В данной программной модели имеются следующие типы заявок:

- создание нового процесса;
- создание дочернего процесса;
- завершение процесса;
- запрос на ввод/вывод;
- завершение ввода/вывода;
- передача управления операционной системе;
- истечение кванта времени.

В центральной части окна находится список процессов, который содержит следующую информацию:

- состояние процесса; отображается цветом строки, значком, а также буквой в столбце «Состояние»:
  - «E» – процесс выполняется;
  - «A» – процесс готов к выполнению;
  - «W» – процесс обратился к устройствам ввода/вывода и ожидает завершения обмена.
- идентификатор процесса – расположен в столбце «PID»;
- идентификатор родительского процесса – расположен в столбце «PPID»;
- текущий приоритет процесса (в системе Windows NT) – расположен в столбце «Приоритет»;
- базовый приоритет процесса (в системе Windows NT) – расположен в столбце «Приоритет»;
- время выполнения процесса (инкрементируется только когда процесс выполняется) – находится в столбце «t»;

– предполагаемое время выполнения процесса – находится в столбце «T»;

– разность между предполагаемым временем и реальным временем выполнения процесса (по сути – время, оставшееся до завершения процесса) – находится в столбце «T-t»; если значение отрицательное (процесс превысил лимит времени), вместо числа выводится «-».

Справа от списка процессов находятся два окна очередей. В любом из них можно отобразить любую очередь (с номерами от 1 до 15). Для добавления значения в конец очереди, необходимо ввести его в нижнее окно ввода и нажать на нижнюю кнопку со стрелкой. Для извлечения значения из очереди необходимо нажать на верхнюю кнопку со стрелкой. Извлеченное значение помещается в окно ввода над кнопкой. В левом окне отображения очереди имеется возможность изменять порядок элементов в очереди путем перетаскивания мышью.

Пользователь имеет возможность выполнить следующие действия:

– ответить на запрос отказом, сразу нажав на кнопку "Подтвердить";

– добавить процесс в список процессов – щелкнуть правой кнопкой мыши на списке процессов и выбрать пункт «Добавить» в контекстном меню, после чего ввести параметры создаваемого процесса: идентификатор, идентификатор родителя (если родителя нет – -1), значение базового и текущего приоритетов, предполагаемое и текущее время выполнения, состояние процесса, и нажать кнопку «OK»;

– добавить идентификатор процесса в очередь;

– поменять порядок элементов в очереди;

– удалить процесс из списка процессов и из очереди – щелкнуть правой кнопкой мыши на нужном элементе в списке процессов и в проявившемся меню выбрать пункт «Удалить»;

– переключить процесс в состояние ожидания – щелкнуть правой кнопкой мыши на нужном элементе в списке процессов и в проявившемся меню выбрать пункт «Переключить в состояние ожидания»;

– переключить процесс в состояние готовности – щелкнуть правой кнопкой мыши на нужном элементе в списке процессов и в проявившемся меню выбрать пункт «Переключит в состояние готовности»;

– выбрать процесс для выполнения – щелкнуть правой кнопкой мыши на нужном элементе в списке процессов и в проявившемся меню выбрать пункт «Переключиться»;

– подтвердить действия, нажав на кнопку "Подтвердить".

В правой части экрана находится список выполненных шагов, которые хранят историю действий пользователя. Элемент в списке помечается красным цветом, если действия пользователя на данном шаге были некорректны. При наведении указателя мыши на элемент списка отображается информация о событии и действиях пользователя на данном шаге. Для отмены произвольного числа шагов необходимо щелкнуть мышью на элемент списка, до которого необходимо очистить историю. Отменённые шаги помечаются серым цветом. До тех пор, пока не выполнено какое-либо действие в окне диспетчера, можно восстановить действия пользователя, щелкнув на записи, соответствующей отмененному шагу.

В нижней части окна расположена строка статуса, в которой отображается общее число ошибок, которые допустил пользователь в процессе выполнения лабораторной работы.

По достижении заданного числа шагов выводится сообщение о выполнении лабораторной работы, а также количество неисправленных ошибок.

1.1.3 Используемые в программной модели дисциплины планирования

Все используемые дисциплины планирования можно разделить на две группы: вытесняющие и невытесняющие.

К первой группе относятся следующие дисциплины:

– FCFS (First Come First Served). Бесприоритетная дисциплина планирования, в которой задачи обслуживаются в порядке их поступления. Используется две очереди: одна – для новых потоков, другая – для потоков, уже использовавших процессорное время. Предпочтение отдается потокам из второй очереди;

– SJN (Shortest Job Next). Используется одна очередь, в которой потоки упорядочены по заявленному времени выполнения. Приоритет отдается потоку с наименьшим заявленным временем. Потоки, превысившие данное ограничение, остаются в конце очереди;

– SRT (Shortest Remaining Time). Аналогична SJN, за исключением того, что потоки в очереди упорядочены по оставшемуся времени выполнения;

Ко второй группе относятся:

– RR (Round Robin). Используется одна очередь, которая заполняется потоками в порядке поступления. Каждому потоку выделяется фиксированный промежуток времени – квант, в течение которого он может выполняться на процессоре. По истечении кванта времени поток добавляется в конец очереди;

– WinNT. Используется 16 очередей (по количеству приоритетов). У каждого потока есть два приоритета: текущий и базовый. Базовый задается при создании потока (процесса) и не меняется, текущий меняется со временем, но не может быть ниже базового. При добавлении процесса в очередь (только в случае, если истек квант времени) приоритет потока уменьшается на

единицу. По завершении операций ввода/вывода ожидающему потоку назначается прибавка к приоритету.

– Unix. Используется 16 очередей (по количеству приоритетов). Имеется два класса задач: со статическими приоритетами (8-15) и динамическими (0-7). Чем дольше процесс с динамическим приоритетом работает без выполнения операций ввода/вывода, тем меньше становится его приоритет. По завершении операций ввода/вывода у такого процесса увеличивается приоритет на 1. При создании процесса или при завершении ввода/вывода процесса с более высоким приоритетом, чем у активного, переключение на данный процесс не происходит. При истечении кванта времени или передаче управления ОС процессорное время выделяется потоку с наивысшим приоритетом.

1.1.4 Недостатки

В текущей программной модели были найдены следующие недостатки:

– приложение доступно только для ОС Windows. Пользователи других операционных систем должны запускать Windows в виртуальной машине либо использовать другие средства по запуску Windows-приложений в других ОС;

– нестабильность. В ходе выполнения лабораторной работы программная модель несколько раз аварийно завершалась, из-за чего результаты выполненной работы безвозвратно терялись;

– ошибки при проверке пользовательских действий. В некоторых случаях было замечено, что программная модель принимала правильную последовательность действий как ошибочную. Это в совокупности с нестабильностью программы усложняет изучение студентами программной модели и, как следствие, увеличивает время на выполнение лабораторной работы;

– нечеткость, «размытость» интерфейса на дисплеях со сверхвысоким разрешением (HiDPI);

– в связи с утерей исходного кода программы и сложностью дизассемблирования определить формат файла задания не представляется возможным, что препятствует разработке новых вариантов заданий.

1.2 Актуальность разработки

Лабораторная работа по данной теме (управление памятью в ОС) имеет важную роль в закреплении лекционного материала и предоставляет студентам возможность изучить более подробно устройство ОС.

Из-за того, что исходный код программной модели утерян, исправить ошибки и недостатки в текущей модели не представляется возможным. Поэтому было принято решение разработать новую программную модель, повторяющую функционал текущей, в которой будут исправлены вышеописанные ошибки и недостатки.

1.3 Техническое задание

1.3.1 Наименование программы

Наименование программы – "Модель диспетчера процессов операционный системы".

1.3.2 Краткая характеристика области применения

Программа предназначена для закрепления студентами лекционного материала по дисциплине «Операционные системы», а именно: управление процессами в операционных системах.

1.3.3 Назначение разработки

Функциональным назначением программы является предоставление студентам возможности изучить работу диспетчера процессов во время выполнения лабораторных работ.

Программа должна эксплуатироваться на ПК студентов, преподавателей и на ПК, установленных в учебных аудиториях Вятского государственного университета. Особые требования к конечному пользователю не предъявляются.

1.3.4 Требования к программе

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1) функции генерации задания;
2) функции сохранения текущего прогресса выполнения задания в файл;
3) функции загрузки задания с сохраненным прогрессом выполнения из файла;
4) функции подсчета количества ошибок, сделанных в ходе выполнения задания;
5) функции просмотра и отмены действий, выполненных в ходе прохождения задания.

Надежное (устойчивое) выполнение программы должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств;
2) использованием лицензионного программного обеспечения.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя:

1) x86-совместимый процессор с тактовой частотой не меньше 1.0 ГГц;
2) дисплей с разрешением не меньше, чем 1024x768;
3) не менее 1 гигабайта оперативной памяти;
4) не менее 100 мегабайт свободного дискового пространства;
5) клавиатура, мышь.

Системные программные средства, используемые программой, должны быть представлены следующими операционными системами:

– 64-разрядная ОС Windows 7/8/8.1/10;
– 64-разрядная ОС Ubuntu 18.04 и выше.

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса и предоставлять возможность выполнять наиболее частые операции с помощью сочетаний клавиш на клавиатуре.

1.3.5 Требования к программной документации

Состав программной документации должен включать в себя:

1) техническое задание;
2) программу и методики испытаний;
3) руководство пользователя;
4) техническую документацию;

5) исходный код.

1.3.6 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) разработка технического задания;
2) рабочее проектирование;
3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;
2) разработка программной документации;
3) испытания программы.

На стадии внедрения должен быть выполнен этап подготовки и передачи программы Заказчику.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;
2) определение и уточнение требований к техническим средствам;
3) определение требований к программе;
4) определение стадий, этапов и сроков разработки программы и документации на неё;
5) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1) разработка, согласование, утверждение программы и методики испытаний;
2) проведение приемо-сдаточных испытаний;
3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена подготовка и передача программы и программной документации в эксплуатацию Заказчику.

2 Разработка структуры приложения

На данном этапе работы необходимо в соответствии с требованиями, поставленными в техническом задании, разработать алгоритмы функционирования и модульную структуру приложения.

2.1 Разработка алгоритмов функционирования

В данной программной модели состояние системы определяется списком процессов и состоянием очередей процессов.

Во время обработки заявки пользователь выполняет некоторую последовательность действий из шагов определенного типа. Данные шаги можно представить как операции, применяемые к состоянию системы.

Для проверки действий, выполненных пользователем над состоянием системы, необходимо разработать алгоритмы обработки поступающих заявок.

2.1.1 Алгоритмы обработки поступающих заявок

Алгоритм обработки заявки «Создание нового процесса»:

1) сначала нужно удостовериться в том, что процесс с заданным PID не существует. Если такой процесс уже есть, то завершить алгоритм;
2) создать процесс с заданными значениями;
3) добавить процесс в соответствующую очередь;
4) переключить процесс в состояние «Готов к исполнению»;
5) выбрать процесс для исполнения согласно дисциплине;
6) если процесса для выполнения нет, то завершить алгоритм; в противном случае перейти к следующему пункту;

7) если на данный момент исполняющегося процесса нет, то извлечь выбранный для исполнения процесс из очереди и переключить его в состояние «Исполняется» и завершить алгоритм; в противном случае перейти к следующему пункту;
8) если используются относительные приоритеты, то завершить алгоритм; в противном случае перейти к следующему пункту;
9) обработать следующие случаи:
    9.1) у текущего исполняющегося процесса приоритет ниже, чем у выбранного. Исполняющийся процесс добавить в соответствующую очередь и переключить в состояние «Готов к исполнению». Извлечь выбранный для исполнения процесс из очереди и переключить его в состояние «Исполняется»;
    9.2) у текущего исполняющегося процесса приоритет не меньше, чем у выбранного. Переключать процессы не нужно.

Схема алгоритма обработки заявки «Создание нового процесса» представлена на рисунке 2.

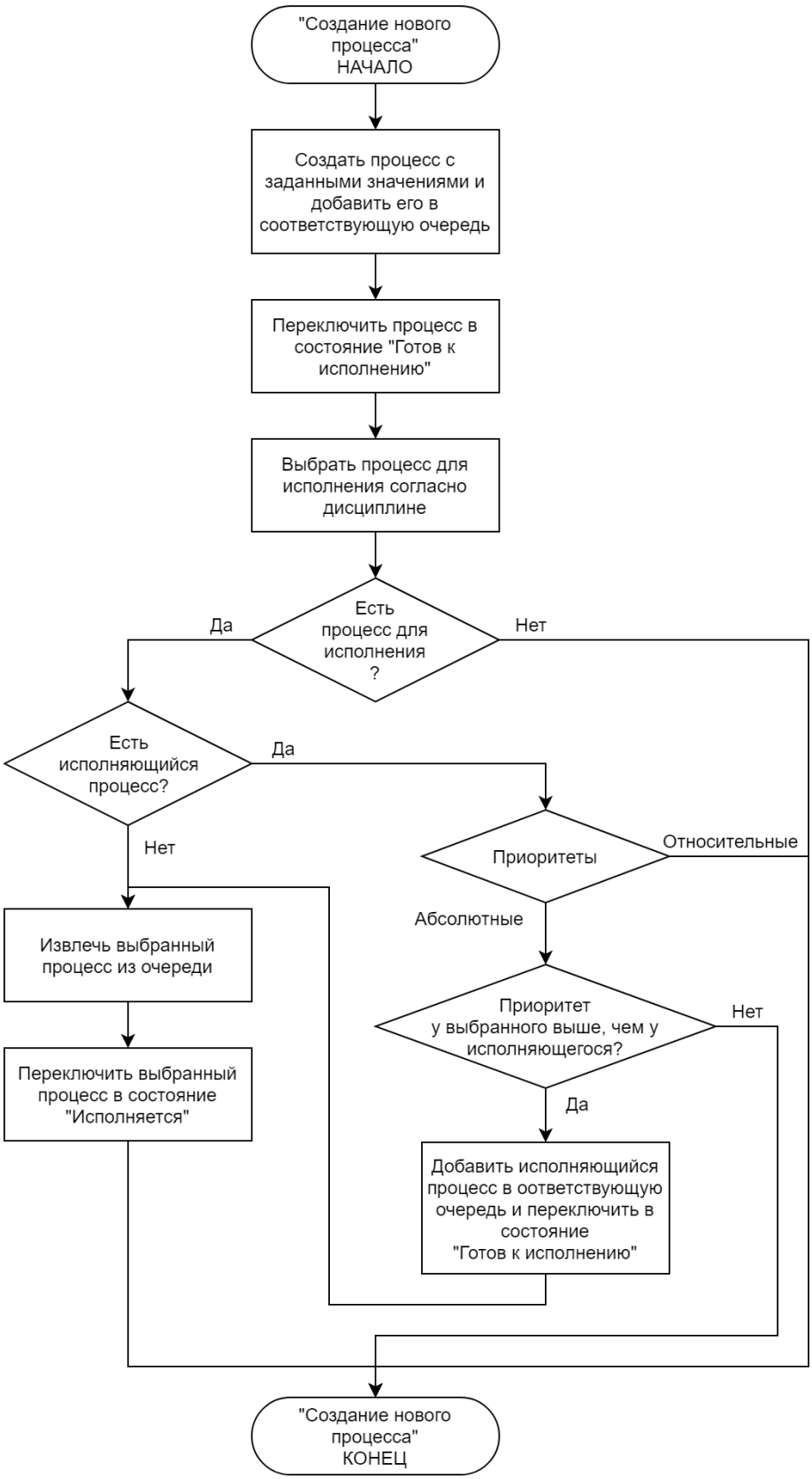


Рисунок 2 – Схема алгоритма обработки заявки «Создание нового процесса»

Алгоритм обработки заявки «Завершение процесса»:

1) сначала нужно удостовериться в том, что процесс с заданным PID существует. Если такого процесса нет, то завершить алгоритм;
2) завершить данный процесс и все его дочерние процессы;
3) если есть исполняющийся процесс, то завершить алгоритм; в противном случае перейти к следующему пункту;
4) выбрать процесс для исполнения согласно дисциплине;
5) если процесса для выполнения нет, то завершить алгоритм; в противном случае перейти к следующему пункту;
6) извлечь выбранный процесс из очереди и переключить его в состояние «Исполняется».

Схема алгоритма обработки заявки «Завершение процесса» представлена на рисунке 3.

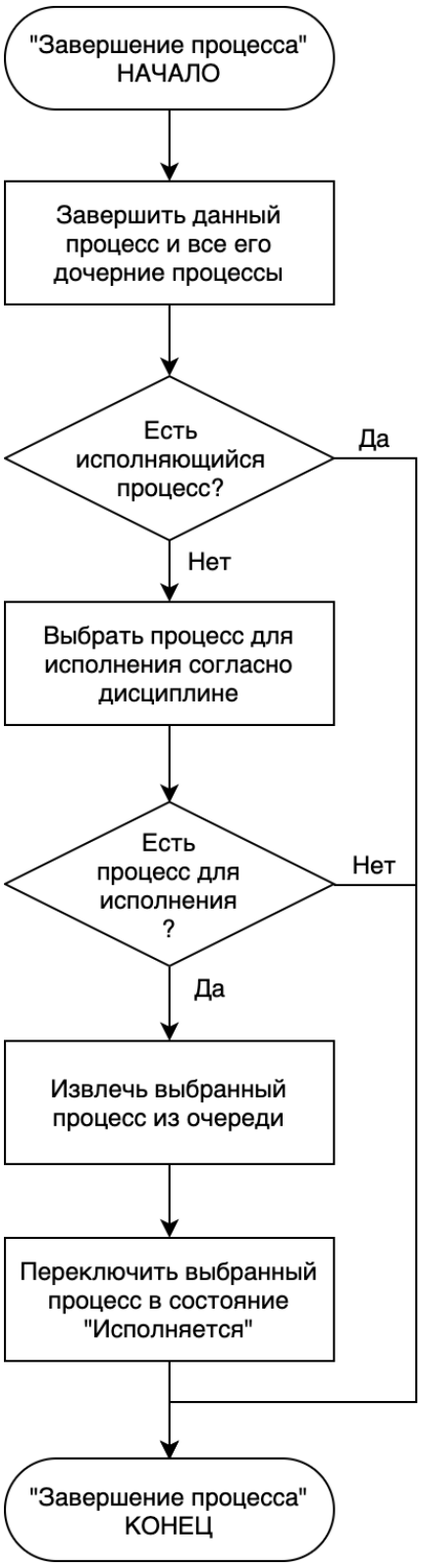


Рисунок 3 – Схема алгоритма обработки заявки «Завершение процесса»

Алгоритм обработки заявки «Запрос на ввод-вывод»:

1) сначала нужно удостовериться в том, что процесс с заданным PID существует и находится в состоянии «Исполняется». Если это не так, то завершить алгоритм;
2) переключить данный процесс в состояние «Ожидание»;
3) выбрать процесс для исполнения согласно дисциплине;
4) если процесса для выполнения нет, то завершить алгоритм; в противном случае перейти к следующему пункту;
5) извлечь выбранный процесс из очереди и переключить его в состояние «Исполняется».

Схема алгоритма обработки заявки «Запрос на ввод-вывод» представлен на рисунке 4.

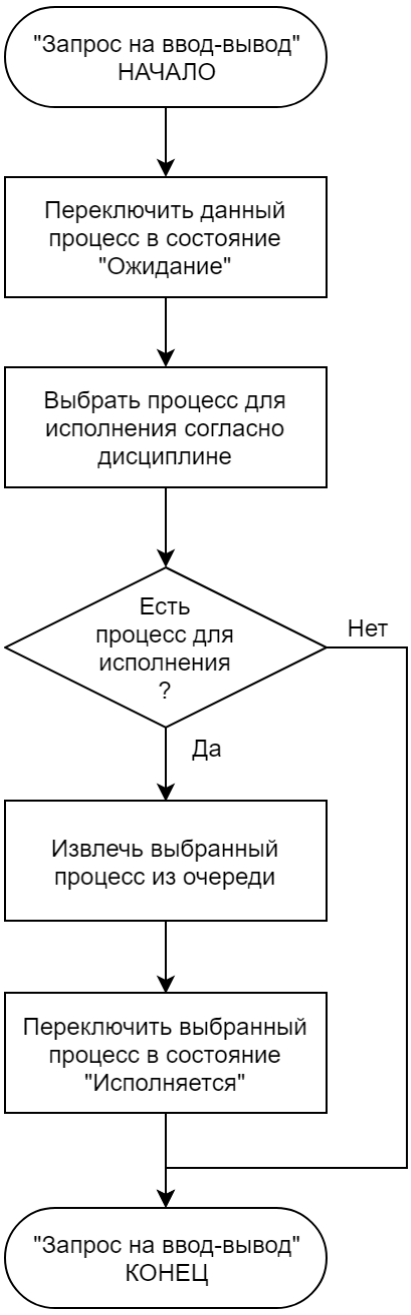


Рисунок 4 – Схема алгоритма обработки заявки «Запрос на ввод-вывод»

Алгоритм обработки заявки «Завершение ввода-вывода»:

1) добавить процесс в соответствующую очередь;
2) переключить процесс в состояние «Готов к исполнению»;
3) выбрать процесс для исполнения согласно дисциплине;

4) если процесса для выполнения нет, то завершить алгоритм; в противном случае перейти к следующему пункту;
5) если на данный момент исполняющегося процесса нет, то извлечь выбранный для исполнения процесс из очереди и переключить его в состояние «Исполняется» и завершить алгоритм; в противном случае перейти к следующему пункту;
6) если используются относительные приоритеты, то завершить алгоритм; в противном случае перейти к следующему пункту;
7) обработать следующие случаи:
    7.1) у текущего исполняющегося процесса приоритет ниже, чем у выбранного. Исполняющийся процесс добавить в соответствующую очередь и переключить в состояние «Готов к исполнению». Извлечь выбранный для исполнения процесс из очереди и переключить его в состояние «Исполняется»;
    7.2) у текущего исполняющегося процесса приоритет не меньше, чем у выбранного. Переключать процессы не нужно.

Схема алгоритма обработки заявки «Завершение ввода-вывода» представлен на рисунке 5.

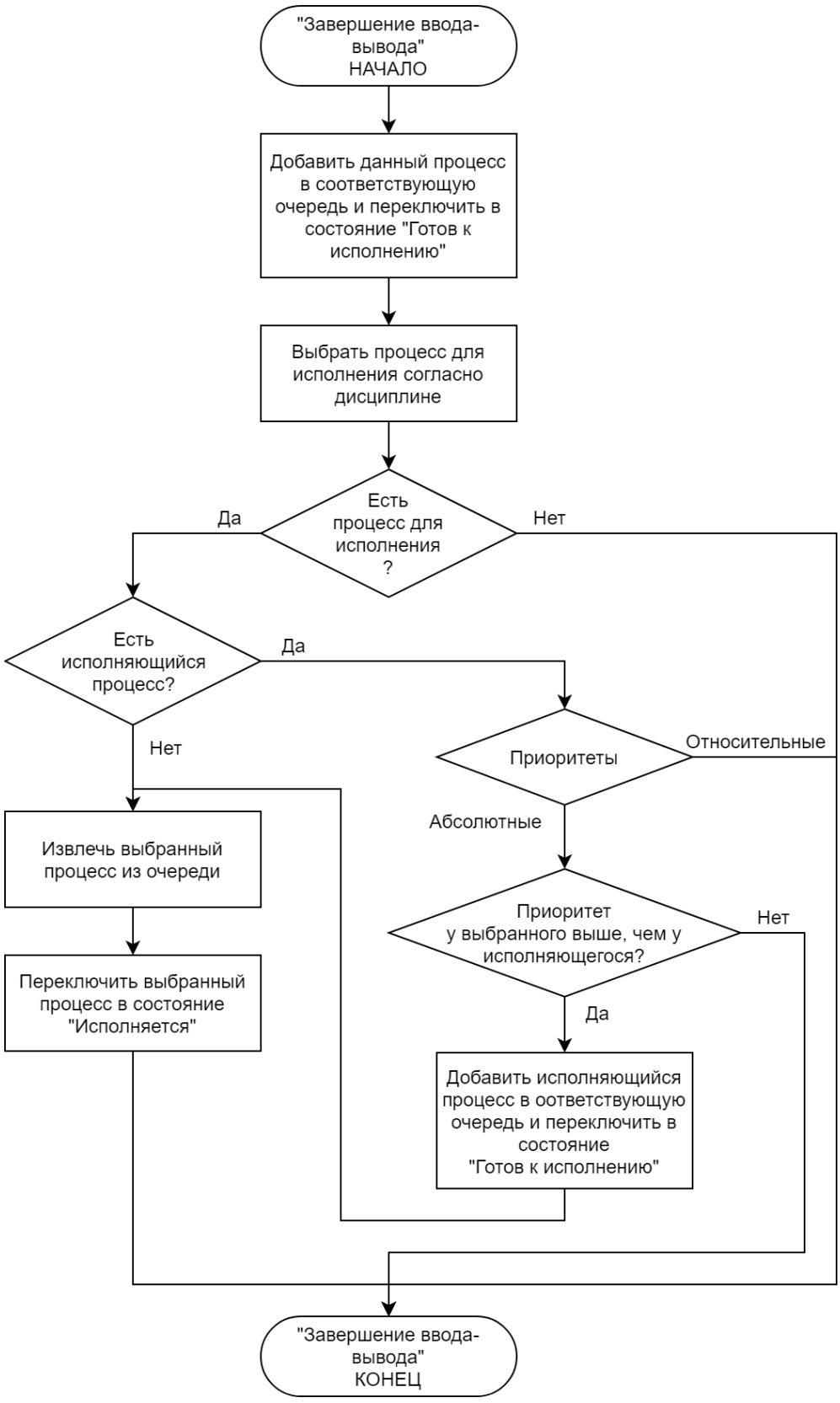


Рисунок 5 – Схема алгоритма обработки заявки «Завершение ввода-вывода»

Алгоритм обработки заявки «Передача управления операционной системе»:

1) добавить процесс в соответствующую очередь;
2) переключить процесс в состояние «Готов к исполнению»;
3) выбрать процесс для исполнения согласно дисциплине;
4) извлечь выбранный процесс из очереди и переключить его в состояние «Исполняется».

Алгоритм обработки заявки «Истечение кванта времени»:

1) добавить процесс в соответствующую очередь;
2) переключить процесс в состояние «Готов к исполнению»;
3) выбрать процесс для исполнения согласно дисциплине;
4) извлечь выбранный процесс из очереди и переключить его в состояние «Исполняется».

2.1.2 Алгоритм генерации заданий

Основная идея алгоритма генерации заданий заключается в том, что каждая следующая заявка создается на основе текущего состояния системы. При этом тип дисциплины, тип заявки, корректность или некорректность заявки, а также параметры заявки определяются случайным образом.

Алгоритм генерации заданий состоит из следующих шагов:

1) выбрать случайным образом дисциплину;
2) создать пустой список заявок задания;
3) на основе текущего состояния системы сгенерировать корректные и некорректные заявки всех типов (по одной на каждый тип);
4) случайным образом определить должна ли текущая заявка быть корректной или нет;

5) если текущая заявка должна быть корректной, то из списка сгенерированных корректных заявок случайным образом выбрать одну и добавить ее в список заявок задания. В противном случае использовать список сгенерированных некорректных заявок;
6) если сгенерировано достаточное количество заявок, то завершить алгоритм, в противном случае перейти к пункту 7;
7) обновить текущее состояние системы в соответствии с выбранной заявкой и дисциплиной, перейти к пункту 3.

Схема алгоритма генерации задания представлена на рисунке 6.

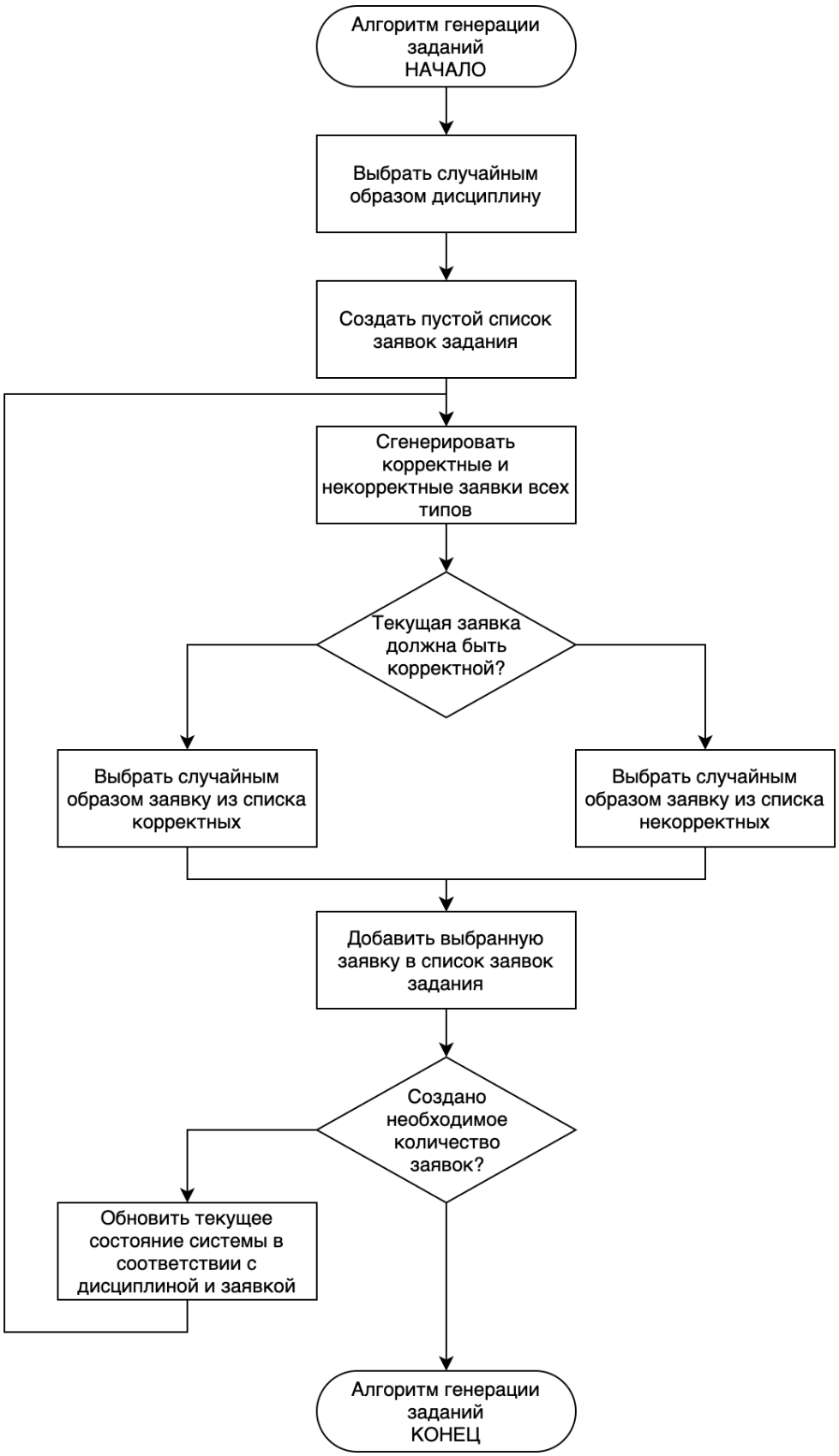


Рисунок 6 – Схема алгоритма генерации задания

2.2 Разработка модульной структуры приложения

Для обеспечения функционирования системы разработана обобщенная структура программного продукта, представляющая собой набор взаимосвязанных модулей, которые реализуют используемые алгоритмы функционирования. Модульная структура приложения представлена на рисунке 7.

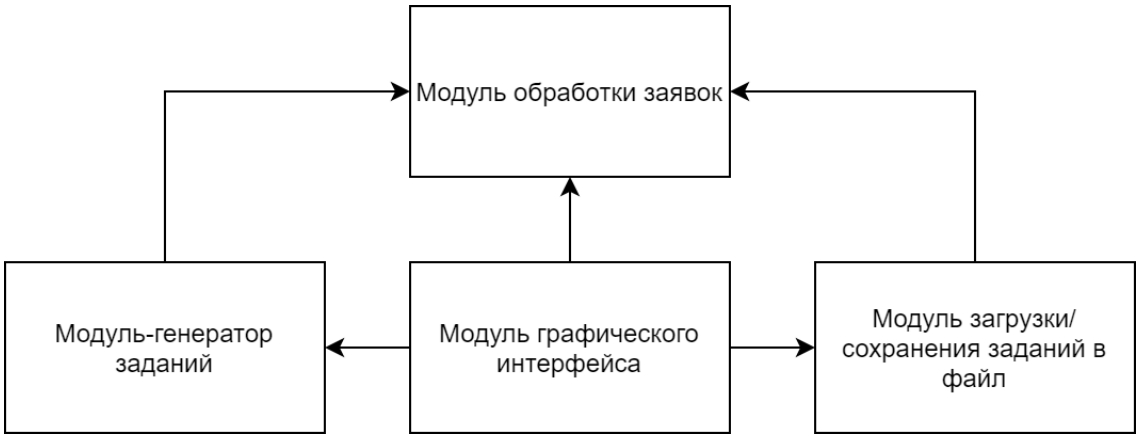


Рисунок 7 – Модульная структура приложения

Каждый из модулей имеет следующее назначение и функционал:

– модуль обработки заявок. Определяет алгоритмы обработки заявок и проверки выполненных пользователем действий, а также определяет необходимые типы данных для представления заявок, состояния процессов и собственно задания на лабораторную работы;

– модуль-генератор заданий. Определяет алгоритмы генерации заданий. Обеспечивает достаточную степень уникальности задания для каждого пользователя без необходимости разработки заданий вручную;

– модуль загрузки и сохранения заданий в файл. Предоставляет возможность сохранять текущее состояние выполняемого задания между запусками приложения, а также обеспечивает защиту от непредвиденного изменения структуры файла;

– модуль графического интерфейса. Является связующим звеном между приложением и пользователем; отображает данные о ходе выполнения задания в текстовом и графическом виде.

3 Программная реализация

На данном этапе работы необходимо выбрать инструменты для кодирования приложения (язык программирования, библиотеки), выполнить реализацию разработанных ранее модулей и алгоритмов, а также разработать графический интерфейс пользователя.

3.1 Выбор инструментов разработки

Так как приложение должно запускаться на различных ОС, а также быть простым в установке и запуске, было принято решение выбрать компилируемый язык программирования с возможностью создания родных для платформы исполняемых файлов. Помимо прочего, для реализации графического интерфейса необходимо выбрать кроссплатформенную библиотеку для его построения.

Среди доступных языков программирования имеются:

– Golang;

– C++;

– Rust;

– C# (.NET Core 3).

Среди кроссплатформенных библиотек для построения графического интерфейса имеются:

– GTK (от сокращения GIMP Toolkit) – кроссплатформенная библиотека элементов интерфейса. Написана на C, доступны интерфейсы для других языков программирования, в том числе для C++, C#, Python. Для проектирования графического интерфейса доступно приложение Glade;

– Qt. Кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Включает в себя все основные классы, которые могут потребоваться при разработке

прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Комплектуется визуальной средой разработки графического интерфейса Qt Designer, позволяющей создавать диалоги и формы в режиме WYSIWYG;

– wxWidgets. Кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений. Основным применением wxWidgets является построение графического интерфейса пользователя, однако библиотека включает большое количество других функций и используется для создания весьма разнообразного ПО. Важная особенность wxWidgets: в отличие от некоторых других библиотек (GTK, Qt и др.), она максимально использует «родные» графические элементы интерфейса операционной системы всюду, где это возможно. Это существенное преимущество для многих пользователей, поскольку они привыкают работать в конкретной среде, и изменения интерфейса программ часто вызывают затруднения в их работе. Написана на C++. Для проектирования графического интерфейса доступно приложение wxGlade;

– AvaloniaUI. Кроссплатформенный XAML-фреймворк для построения графических интерфейсов пользователя для платформ .NET Framework, .NET Core и Mono.

При этом нужно учитывать, что не все комбинации языков программирования с библиотеками графического интерфейса возможны, удобны в разработке и достаточно стабильны. Для языков Golang и Rust не имеются стабильные версии перечисленных библиотек, для GTK сборка C++-приложений под Windows требует нетривиальной настройки окружения вплоть до эмуляции Linux-окружения в Windows, а AvaloniaUI доступна только под C# и не имеет еще стабильной версии. Среди доступных вариантов остаются:

– C++ и wxWidgets;

– C# и GTK;

– C++ и Qt.

В ходе ознакомления с библиотекой wxWidgets были выявлены следующие недостатки: недостаточно подробная документация, нетривиальная настройка режима Drag’n’Drop (необходим для перемещения элементов ГПИ, представляющих блоки памяти), скудный функционал конструктора форм wxGlade.

У GTK графические элементы выглядят достаточно непривычно в сравнении с родными для Windows приложениями; имеет те же недостатки, что и wxWidgets.

По степени интеграции выигрывает Qt со средой разработки QtCreator.

В качестве системы сборки выбрана CMake – одна из наиболее популярных среди проектов, написанных на C++.

3.2 Реализация модулей приложения

3.2.1 Модуль обработки заявок

В данном модуле должны быть реализованы алгоритмы обработки заявок, а также определены следующие необходимые типы:

– структура Process. Описывает отдельно взятый процесс. Основными атрибутами являются: PID (тип int), PID родительского процесса, базовый и текущий приоритеты (тип int), состояние (тип int), текущее и заявленное времена выполнения процесса (тип int). Для доступа к атрибутам реализованы методы pid(), ppid() и basePriority(), priority(), timer() и worktime() соответственно;

– структура ProcessesState. Описывает состояние системы. Содержит два атрибута: список процессов (тип std::vector< Process>) и список из 16 очередей процессов (тип std::array<std::deque<int>, 16>);

– структуры, описывающие заявки:
  - CreateProcessReq. Основные атрибуты: PID (тип int), PID родительского процесса (тип int), базовый приоритет (тип int), заявленное время выполнения (тип int);
  - TerminateProcessReq. Атрибуты: PID (тип int);
  - InitIO. Атрибуты: PID (тип int);
  - TerminateIO. Атрибуты: PID (тип int) и augment (тип int). augment используется только в дисциплине WinNT;
  - TransferControl. Атрибуты: PID (тип int);
  - TimeQuantumExpired.

Для обеспечения возможности хранить заявки различного типа в контейнерах, подобных std::vector, был добавлен тип-сумма Request, являющийся типом std::variant.

Дисциплины планирования выполнены в виде отдельных классов, реализующих интерфейс AbstractStrategy, который содержит следующие методы:

– toString() – возвращает строковое обозначение стратегии;

– type() – возвращает числовое обозначение стратегии;

– processRequest(Request, ProcessesState) – обрабатывает заявки всех типов;

– schedule(ProcessesState) – вызывает планировщик.

Диаграмма классов данного модуля представлена на рисунке 8.

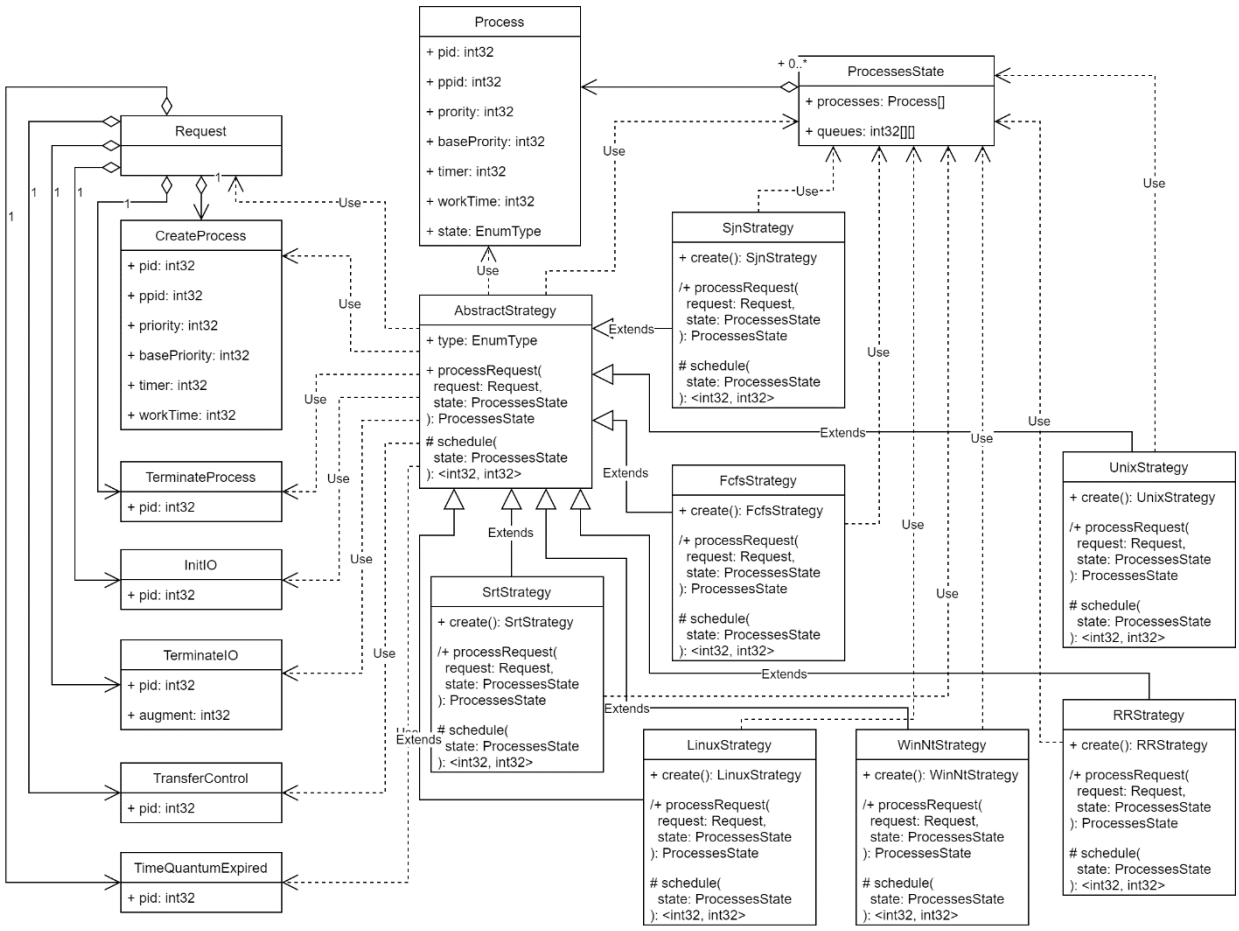


Рисунок 8 – Диаграмма классов модуля обработки заявок

3.2.2 Модуль загрузки и сохранения заданий в файл

Данный модуль реализует функционал сохранения текущего прогресса выполнения задания в файл, загрузку задания из файла с защитой от непредвиденных изменений.

Состояние задания определяется такими параметрами как:

- список всех заявок;
- количество уже выполненных заявок;
- количество допущенных пользователем ошибок;
- тип дисциплины;
- журнал действий пользователя;
- состояние процессов после последней выполненной заявки.

Все перечисленные выше параметры должны быть сохранены в файл. Последний параметр позволяет защитить содержимое файла от изменений, направленных на подделку результатов выполнения задания, потому что для этого атакующему необходимо будет самостоятельно обработать заявки, чтобы получить искомое состояние памяти.

В качестве формата файла задания был выбран JSON, потому что он имеет синтаксис, достаточно удобный как для человека, так и для машины. JSON является текстовым форматом, а не бинарным, поэтому файлы в формате JSON можно просматривать и редактировать в любом текстовом редакторе.

Задания в файле сохранены в виде массива JSON-объектов. Структура объекта задания представлена в таблице 1.

Таблица 1 – Структура объекта задания

Поле	Тип	Описание
type	String	Тип задания. Значение: “PROCESSES_TASK”
strategy	String	Название дисциплины. Допустимые значения: “FCFS”, “SJN”, “SRT”, “ROUNDROBIN”, “UNIX”, “WINNT”, “LINUXO1”
completed	Number	Количество обработанных заявок
fails	Number	Количество допущенных пользователем ошибок
state	Object	Объект, описывающий состояние процессов
actions	Array	Массив строк, содержащих информацию о действиях пользователя для каждой заявки
requests	Array	Массив заявок, которые диспетчер должен обработать

В объекте, описывающем состояние процессов (поле “state”), хранится список всех процессов, а также содержимое всех очередей. Структуры

объектов состояния процессов, дескриптора процесса и заявки представлены в таблицах 2, 3 и 4 соответственно.

Таблица 2 – Структура объекта состояния процессов

Поле	Тип	Описание
processes	Array	Массив из дескрипторов процессов
queues	Array	Массив из 16 очередей, каждая - массив из идентификаторов процессов

Таблица 3 – Структура объекта дескриптора процесса

Поле	Тип	Описание
pid	Number	PID процесса
ppid	Number	PID родительского процесса или -1
priority	Number	Приоритет
basePriority	Number	Базовый приоритет
timer	Number	Время работы
workTime	Number	Заявленное время работы
state	String	Состояние процесса. Допустимые значения: “ACTIVE”, “EXECUTING”, “WAITING”

Таблица 4 – Структура объекта заявки

Заявка на создание процесса		
Поле	Тип	Описание
type	String	Тип заявки. Значение: “CREATE_PROCESS”
pid	Number	PID процесса
ppid	Number	PID родительского процесса или -1
priority	Number	Приоритет
basePriority	Number	Базовый приоритет
timer	Number	Время работы
workTime	Number	Заявленное время работы
Заявка на завершение процесса		
Поле	Тип	Описание
type	String	Тип заявки. Значение: “TERMINATE_PROCESS”
pid	Number	PID процесса
Заявка на инициализацию ввода/вывода		
Поле	Тип	Описание
type	String	Тип заявки. Значение: “INIT_IO”
pid	Number	PID процесса
Заявка на завершение ввода/вывода		
Поле	Тип	Описание
type	String	Тип заявки. Значение: “TERMINATE_IO”
pid	Number	PID процесса
augment	Number	Прибавка к текущему приоритету
Заявка на передачу управления операционной системе		
type	String	Тип заявки. Значение: “TRANSFER_CONTROL”
pid	Number	PID процесса
Заявка на обработку события "Истек квант времени"		
type	String	Тип заявки. Значение: “QUANTUM_EXPIRED”

Пример содержимого файла задания представлен на рисунке 9.


```
[
  {
    "completed": 2,
    "fails": 0,
    "actions": [],
    "requests": [
      {
        "pid": 11, "ppid": -1, "priority": 0,
        "basePriority": 0, "timer": 0,
        "type": "CREATE_PROCESS", "workTime": 0
      },
      {
        "pid": 6, "ppid": 11, "priority": 0,
        "basePriority": 0, "timer": 0,
        "type": "CREATE_PROCESS", "workTime": 0
      }
    ],
    "state": {
      "processes": [
        {
          "pid": 6, "ppid": 11, "priority": 0,
          "basePriority": 0, "state": "ACTIVE",
          "timer": 0, "workTime": 0
        },
        {
          "pid": 11, "ppid": -1, "priority": 0,
          "basePriority": 0, "state": "EXECUTING",
          "timer": 2, "workTime": 0
        }
      ],
      "queues": [
        [6], [], [], [], [], [], [], [],
        [], [], [], [], [], [], [], []
      ]
    },
    "strategy": "ROUNDROBIN",
    "type": "PROCESSES_TASK"
  }
]
```


Рисунок 9 – Пример содержимого файла задания

Загрузка заданий из файла реализована в функции loadTasks(), сохранение – saveTasks().

Диаграмма классов данного модуля представлена на рисунке 10.

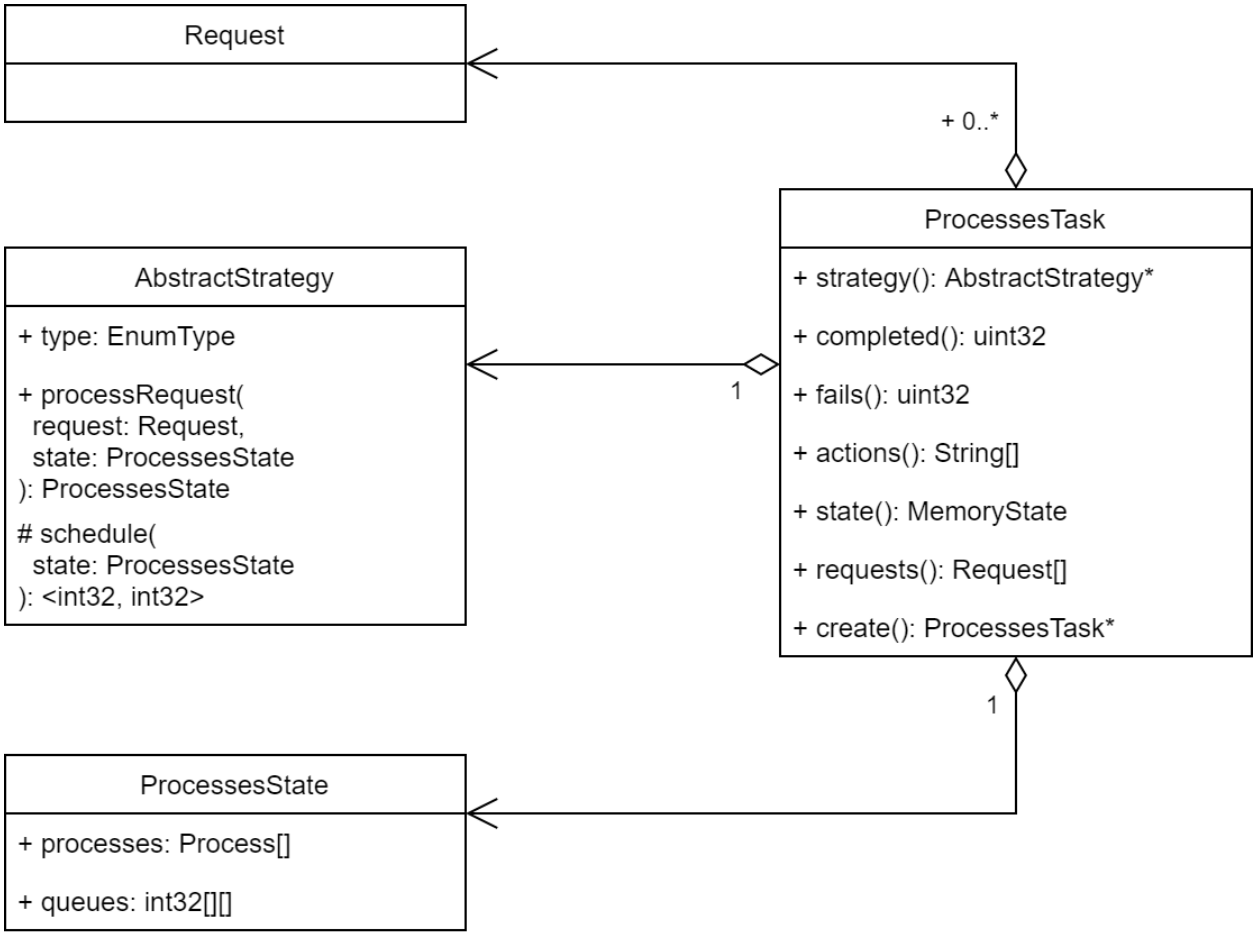


Рисунок 10 – Диаграмма классов модуля загрузки и сохранения задания в файл

3.2.3 Модуль-генератор заданий

Данный модуль исполнен в виде набора функций и классов, реализующих алгоритм генерации заданий для каждой стратегии в отдельности с вынесением общих частей в базовый класс AbstractTaskGenerator, содержащий следующие методы:

– basis(ProcessesState, bool) – создает экземпляры заявок (корректные или некорректные) в определенном соотношении по типам. Соотношение между типами заявок зависит количества процессов, находящихся в различных состояниях;

– applyFilters(std::vector<Requests>, Request, bool) – исключает вырожденные случаи следующих типов:

  – 2 подряд идущие заявки TimeQuantumExpired или TransferControl;
  – после заявки CreateProcessReq идет заявка TransferControl или TerminateProcessReq с таким же PID;
  – после InitIO идет заявка TerminateIO с таким же PID;
  – первая в списке заявка не является CreateProcessReq;

– generate(ProcessesState, Request, bool) – возвращает список из сгенерированных заявок для выбора одной из них.

Были разработаны следующие функции:

– randStrategy() – выбирает случайным образом дисциплину планирования;

– collectStats(ProcessesState) – собирает статистику по количеству процессов, находящихся в состояниях ожидания, исполнения и готовности;

– generate() – использует описанные выше функции и классы генераторов для построения задания. Количество заявок по умолчанию – 40.

Опытным путем были подобраны следующие соотношения:

– соотношение между количеством некорректных и корректных заявок – 1 : 7;

– соотношения между количеством заявок различных типов, генерируемых в функции basis:

- CreateProcessReq : TerminateProcessReq : InitIO : TerminateIO : TransferControl = 2 : 3 : 1 : 1 : 2;
- если процессов, находящихся в состоянии готовности меньше 5, то добавляются 2 заявки CreateProcessReq;
- если процессов, находящихся в состоянии готовности больше 7, то добавляются 3 заявки TerminateProcessReq;
- если процессов, находящихся в состоянии ожидания меньше 1, то добавляются 2 заявки InitIO;
- если процессов, находящихся в состоянии ожидания больше 3, то добавляются 2 заявки TerminateIO;
- для вытесняющих дисциплин добавляются 2 заявки TimeQuantumExpired;
- в 3 из 5 случаев заявки CreateProcessReq соответствуют созданию дочерних процессов.

Также в генераторе установлено ограничение на максимальное количество процессов – 24.

3.2.4 Модуль графического интерфейса

Проанализировав интерфейс предыдущей установки, было принято решение внести в него несколько небольших изменений:

– добавить кнопку «Сбросить». При нажатии на нее все изменения, сделанные в ходе обработки текущей заявки, будут сброшены;

– вместо пиктограмм на кнопках будут содержаться соответствующие их смыслу надписи;

– для удобства пользования для кнопок «Подтвердить» и «Сбросить» добавлены клавиатурные сочетания «Alt+Enter» и «Ctrl+Z» соответственно;

– для состояний процессов сделаны более заметные пиктограммы в виде кружков разного цвета;

– при попытке закрыть основное окно программы будет выводиться диалог подтверждения, чтобы исключить потерю результатов из-за случайного закрытия программы;

– на случай поиска и устранения непредвиденных неполадок сделано автоматическое сохранение сгенерированного задания в файл во временной папке пользователя. Для доступа к ней в меню приложения добавлен отдельный пункт (рисунок 13).

Экранная форма основного окна программы представлена на рисунке 11.

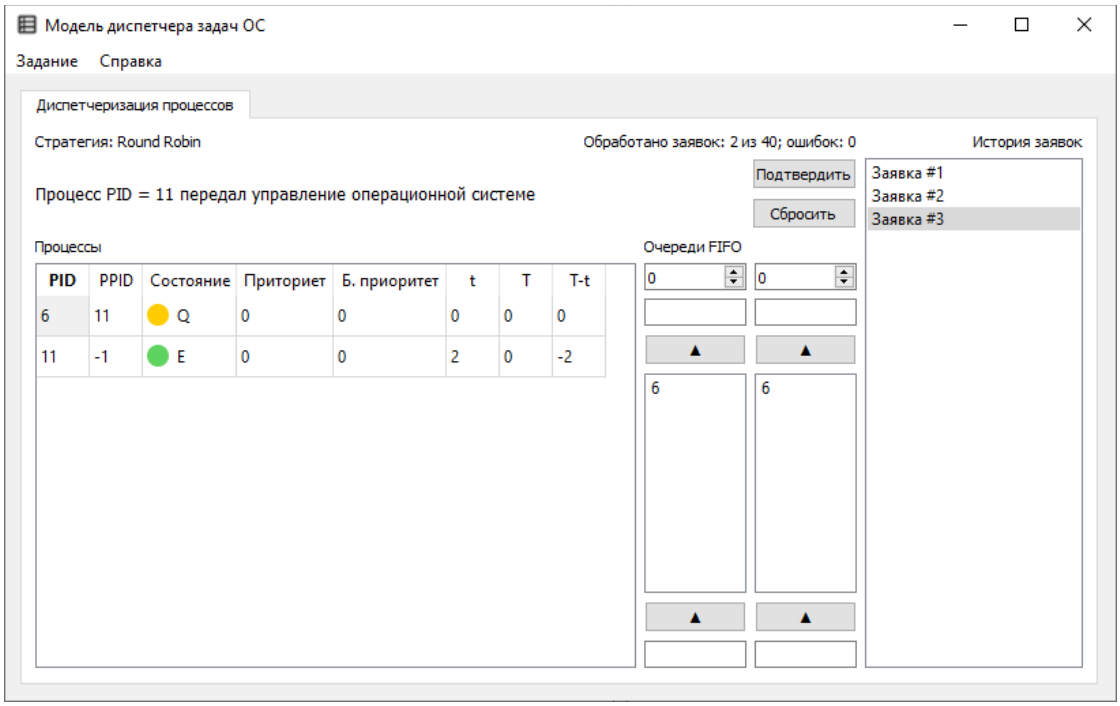


Рисунок 11 – Экранная форма основного окна программы

Диаграмма классов данного модуля представлена на рисунке 12.

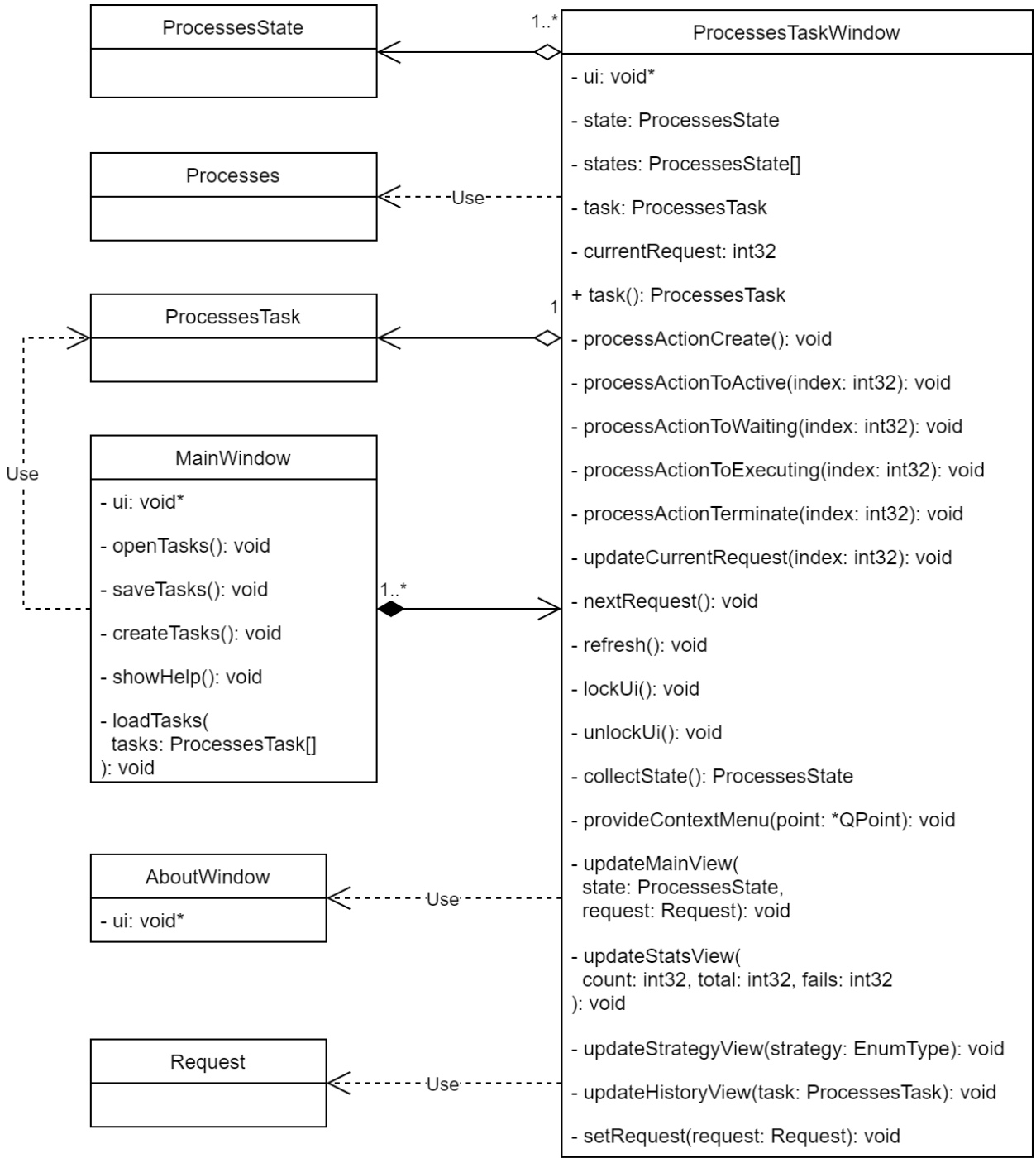


Рисунок 12 – Диаграмма классов модуля графического интерфейса

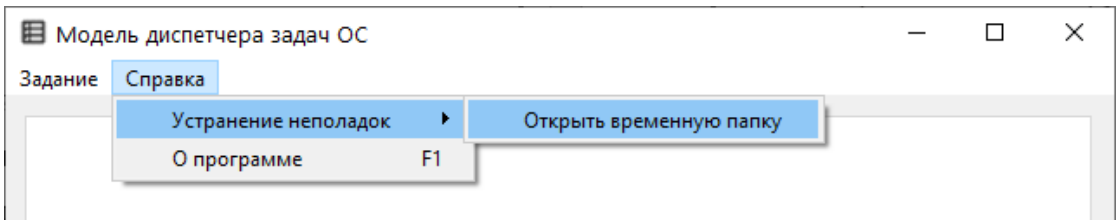


Рисунок 13 – Пункт меню для доступа к временной папке

На основе разработанной структуры приложения и алгоритмов функционирования была выполнена программная реализация приложения. Модули приложения были реализованы в виде классов и функций; в графический интерфейс пользователя были внесены небольшие улучшения, а также был разработан формат файлов заданий. Диаграмма разработанных классов представлена в приложении А.

Заключение

В ходе выполнения курсового проекта было разработано программное обеспечение – лабораторная установка «Модель диспетчера процессов операционной системы». Изначально предполагалось исправление и доработка существующей установки, однако такая задача оказалось крайне сложной в сравнении с разработкой нового приложения.

В качестве направления дальнейшего развития можно выбрать разработку отдельного конструктора заданий для преподавателя и обеспечение шифрования файлов заданий.

Перечень сокращений

ГПИ – графический пользовательский интерфейс

ОС – операционная система

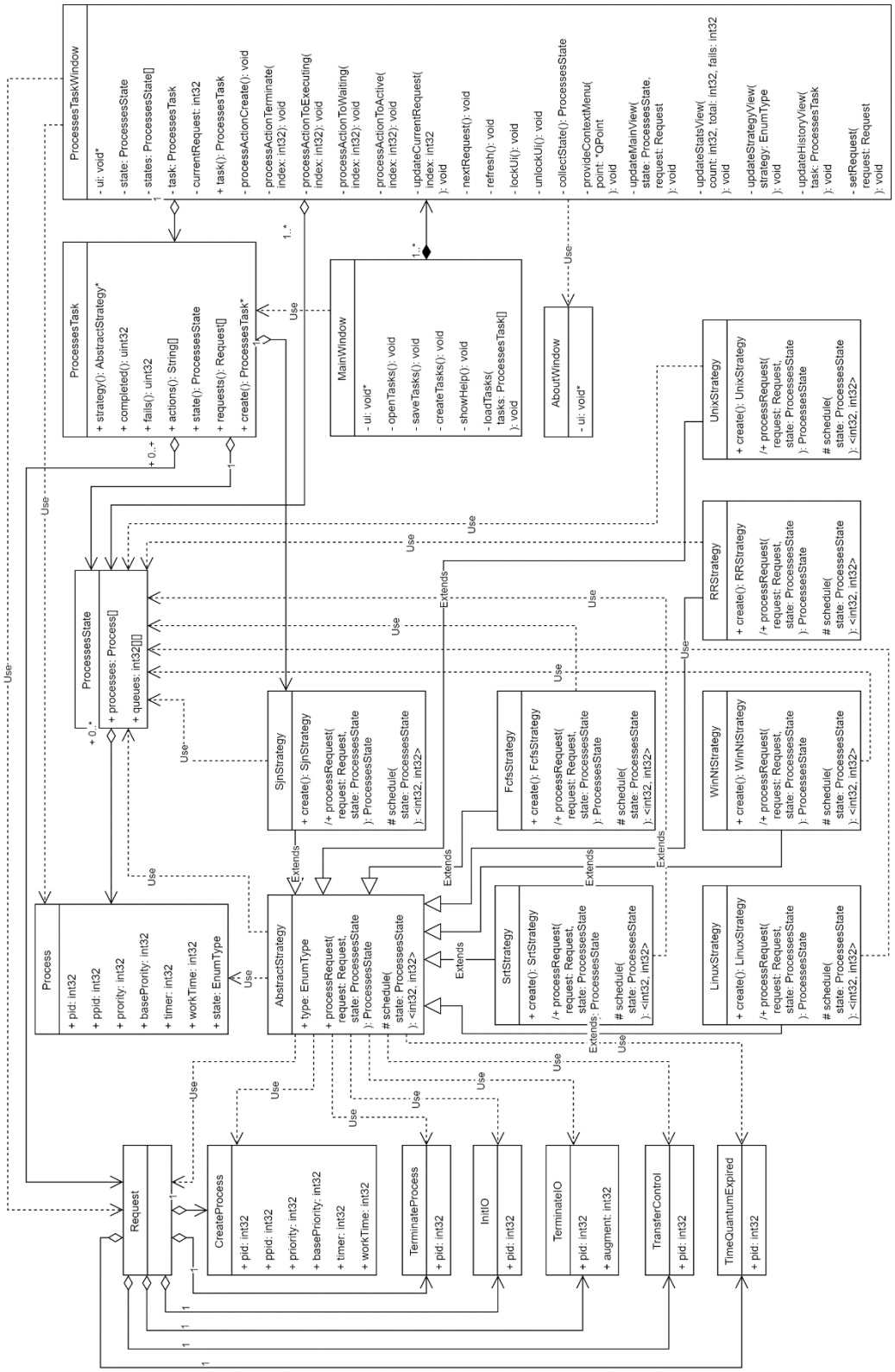
ПК – персональный компьютер

PID – process identifier, идентификатор процесса

Приложение А

(обязательное)

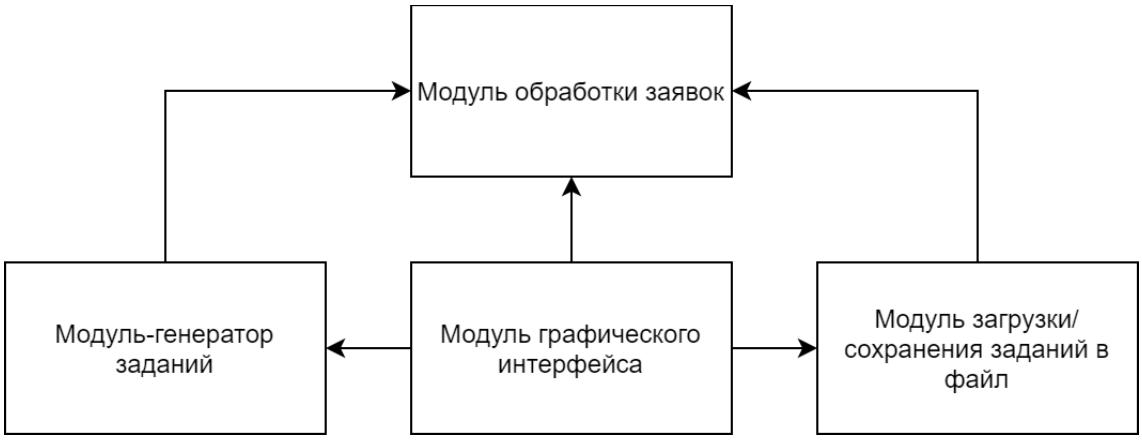
Диаграмма классов



Приложение Б

(обязательное)

Модульная структура приложения



Приложение В

(справочное)

Библиографический список

1 Караваева О. В. Операционные системы. Управление памятью и процессами: учебно-методическое пособие / О. В. Караваева. – Киров: ФГБОУ ВО «ВятГУ», 2016. – 39 с.

2 Avalonia мои за и против / Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/471728/ – Загл. с экрана.

3 Qt Reference Pages | Qt 5.14 [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qt-5/reference-overview.html. – Загл. с экрана. – Англ.

4 AvaloniaUI/Avalonia: A multi-platform .NET UI framework [Электронный ресурс]. – Режим доступа: https://github.com/AvaloniaUI/Avalonia. – Загл. с экрана. – Англ.

5 wxWidgets – Wikipedia [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/WxWidgets. – Загл. с экрана. – Англ.

6 GTK – Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/GTK. – Загл. с экрана. – Англ.

7 Qt – Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Qt. – Загл. с экрана. – Англ.