|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт кибернетики | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

|  |  |
| --- | --- |
| **ОТЧЕТ** | |
| **ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ** | |
|  | |
| **Тема практики «Мониторинг использования ресурсов и состояния вычислительной системы**»  *(наименование темы)* | |
| Студент группы            КМБО-05-19 | *А.А. Булаво* |
| Руководитель практики | *А.А. Свинцова* |

|  |  |  |
| --- | --- | --- |
| «Отчет представлен к рассмотрению» | «\_\_\_» \_\_\_\_\_\_\_\_\_\_2021 г. | *(подпись студента)* |
|  |  |  |
| «Отчет утвержден.  Допущен к защите» | «\_\_\_» \_\_\_\_\_\_\_\_\_\_2021 г. | *(подпись руководителя)* |

Москва 2021

# ОГЛАВЛЕНИЕ

[Введение 3](#_Toc85213996)

[1. Постановка задачи 3](#_Toc85213997)

[1.1 Задание 3](#_Toc85213998)

[1.2 Исходные данные 3](#_Toc85213999)

[1.3 Планируемые результаты 3](#_Toc85214000)

[2. Описание работ и результатов 3](#_Toc85214001)

[2.1 Используемые средства разработки 3](#_Toc85214002)

[2.2 Содержание и результат выполненных работ 4](#_Toc85214003)

[2.3 Доступ к результатам 13](#_Toc85214004)

[3. Заключение 13](#_Toc85214005)

[Список использованных источников 14](#_Toc85214006)

Введение

Одной из задач управления вычислительными кластерами является оптимизация распределения нагрузки между отдельными узлами кластера с целью оптимизации производительности и равномерного распределения износа. Для того, чтобы решить данную задачу, необходимо получить информацию об использовании ресурсов, таких как нагрузка на центральный процессор и загрузка оперативной памяти в количественном отношении, и состоянии отдельного узла в целом, которую можно решить путем использования специального программного обеспечения, разработка которого была дана в качестве задания на практику.

1. Постановка задачи
   1. Задание

Необходимо написать клиент-серверное приложение, выполняющее мониторинг состояния как вычислительного комплекса в целом, так и его отдельных его компонентов, работающее на компьютерах под управлением операционных систем Windows и Linux.

* 1. Исходные данные

Количественные и качественные сведения о нагрузке на вычислительный кластер и состоянии отдельных процессов (запущен/не запущен).

* 1. Планируемые результаты

Графическое приложение, отображающее информацию об узле и отдельных его компонентах.

1. Описание работ и результатов
   1. Используемые средства разработки

Данное приложение реализовано в виде двух модулей – клиента и сервера. Оба модуля имеют графический интерфейс и работают под управлением операционных систем Windows и Linux, также оба модуля активно используют интерфейсы программирования данных операционных систем. Поэтому, в качестве основного языка программирования был выбран C++. В качестве основного фреймворка была выбрана кроссплатформенная библиотека Qt, которая позволяет получить одинаковый графический интерфейс на всех целевых платформах. Также, qt реализован на языке C++, что позволяет подключать сторонние библиотеки, написанные на том же языке, которые могут эффективно взаимодействовать со средствами операционной системы. Помимо кроссплатформенного графического интерфейса, данное приложение также содержит модули, структура которых зависит от архитектуры выбранной операционной системы. Это модули отвечающие за получение сведений о состоянии узла. Для разработки таких модулей в среде Linux также использовался Qt. В среде Windows такие модули были реализованы в виде динамически связываемых библиотек, которые были разработаны с использованием среды разработки Microsoft Visual Studio, и затем подключены в проекте Qt. В качестве основного компилятора был выбран GCC как универсальный кроссплатформенный вариант. Для разработки библиотек проверки состояния узла под Windows использовался компилятор MSVC, версия 2015.

* 1. Содержание и результат выполненных работ

В соответствии с планом выполнения индивидуального задания, была выполнена следующая подготовительная работа:

* Была изучена организация работы в инструментальной среде разработки QtCreator.
* Была изучена технология разработки графических приложений с использованием фреймворка Qt.
* Была изучена технология разработки клиент-серверных приложений с использованием фреймворка Qt.
* Были изучены методы получения информации о состоянии узла и отдельных его компонентов с помощью интерфейса прикладного программирования Windows NT и сопряженных библиотек.
* Были изучены методы получения информации о состоянии узла и отдельных его компонентов с помощью интерфейсов программирования UNIX-подобных операционных систем.

Разработка самого приложения выполнялась поэтапно, во время каждого этапа реализовывались возможности отдельных модулей:

* Сначала была разработана программа, определяющая состояние другой программы. При запуске появляется графический интерфейс (Рисунок 1), который содержит два компонента, кнопка «обзор», при нажатии на которую, открывается диалоговое окно, в котором предлагается выбрать путь интересующему исполняемому файлу (Рисунок 2). После подтверждения выбора путем нажатия кнопки «открыть» в диалоговом окне, в компоненте текстового ввода начинает отображаться о текущем состоянии выбранного приложения (Рисунок 3, Рисунок 4, Рисунок 5), которая обновляется в режиме реального времени.

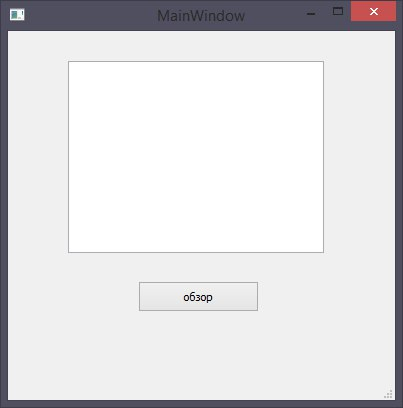


Рисунок 1 – Графический интерфейс программы

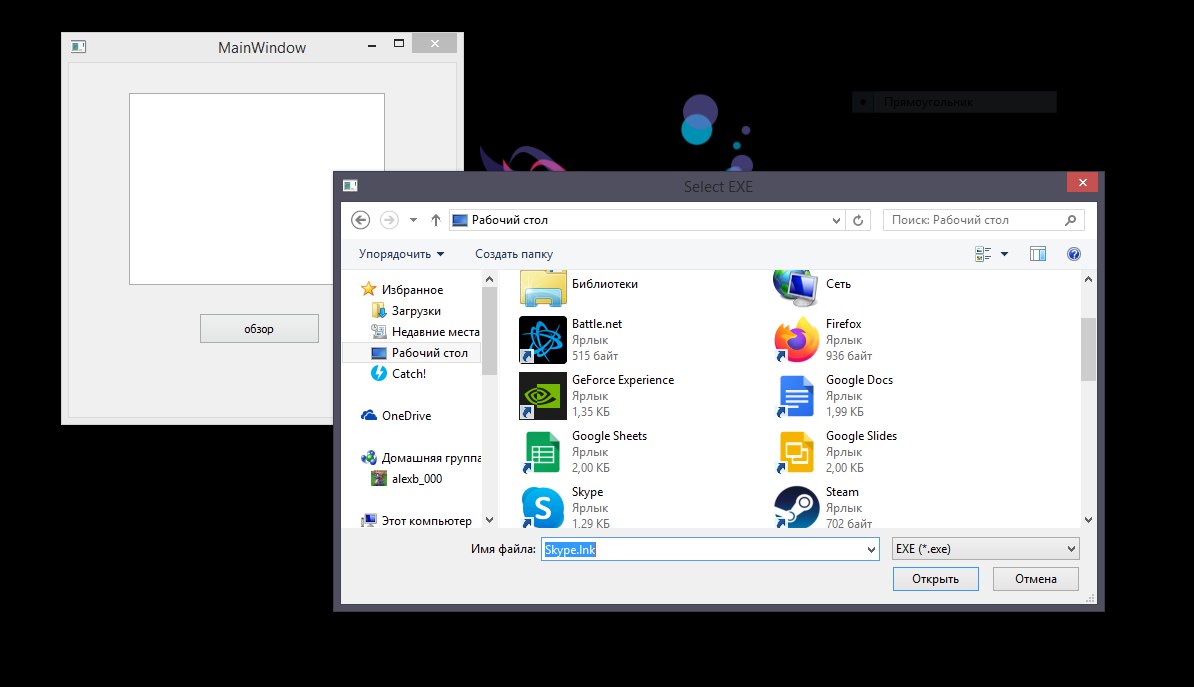


Рисунок 2 – Диалоговое окно выбора исполняемого файла

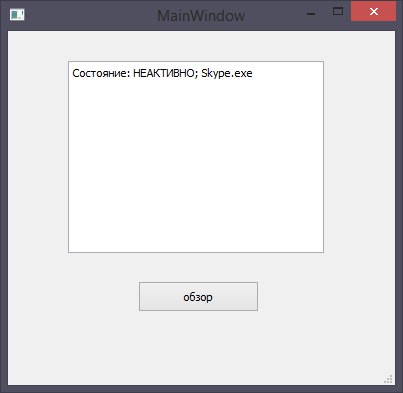


Рисунок 3 – Вывод информации о состоянии процесса, когда он не запущен

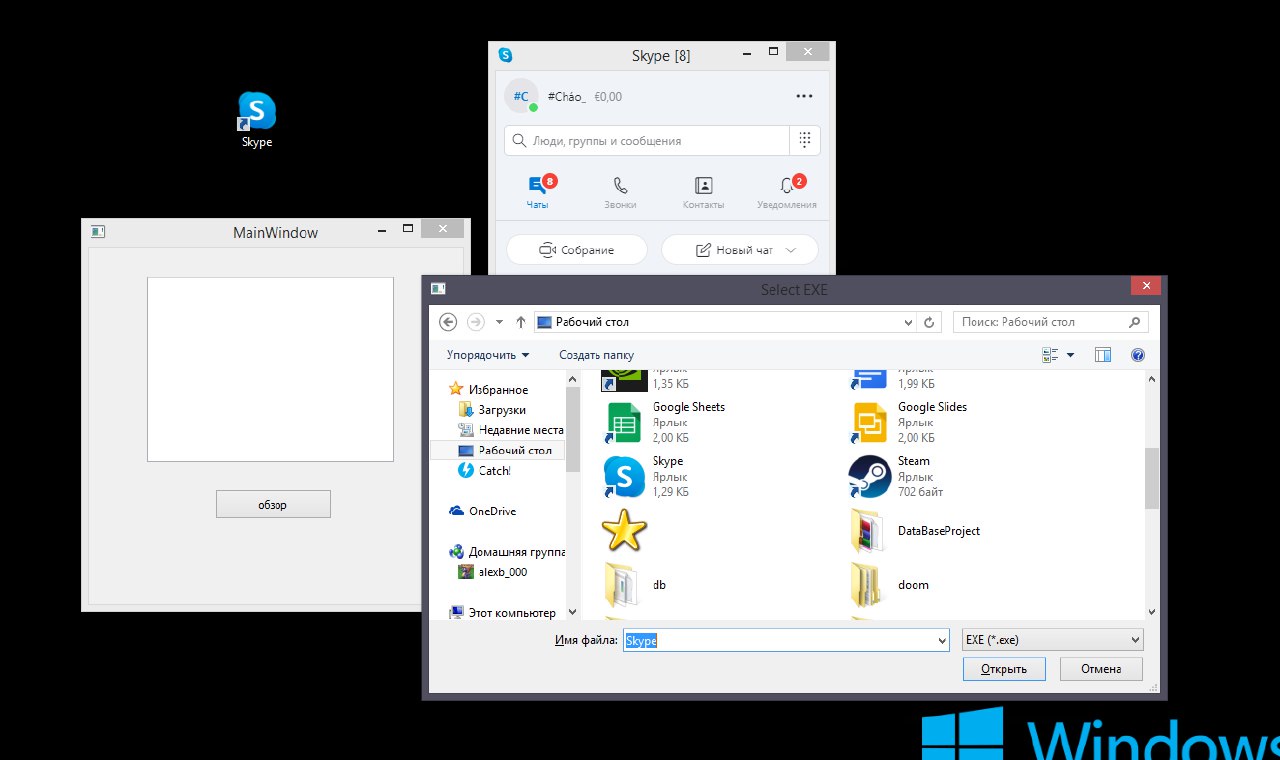


Рисунок 4 – Открытие исполняемого файла

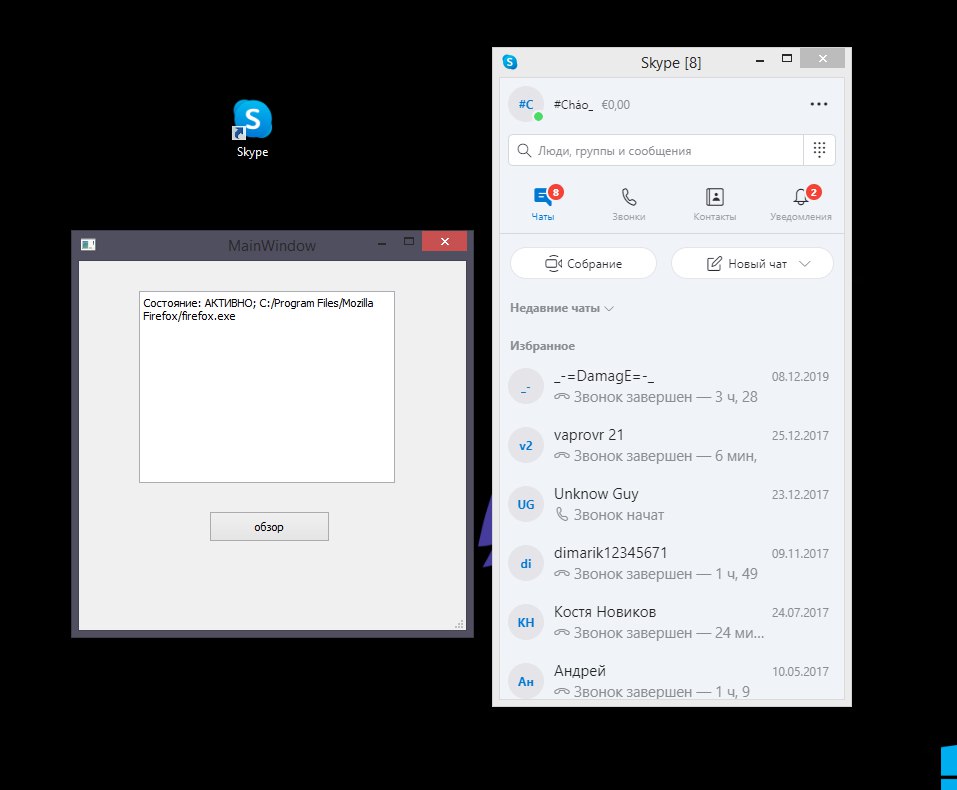


Рисунок 5 - Вывод информации о состоянии процесса, когда он запущен

* Затем, вывод состояния программы был организован через виджет QTableWidget (Рисунок 6) и был реализован такой вариант программы, работающий в среде операционных систем на базе ядра Linux (Рисунок 7, Рисунок 8 и Рисунок 9).

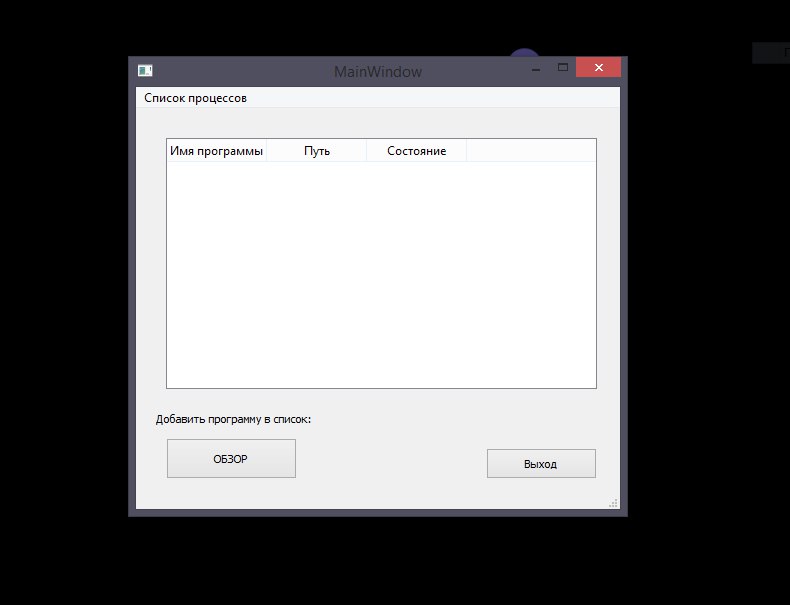


Рисунок 6 – Исправленный и дополненный графический интерфейс программы

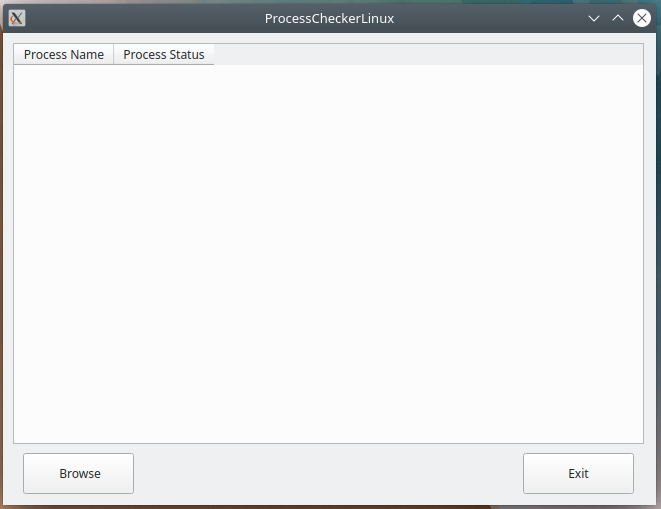


Рисунок 7 – Интерфейс приложения под Linux

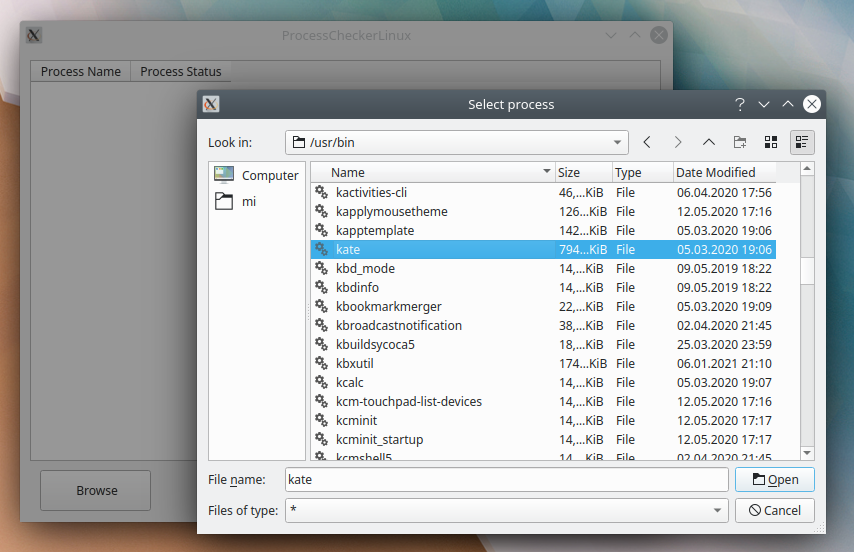


Рисунок 8 – Диалоговое окно выбора исполняемого файла в Linux

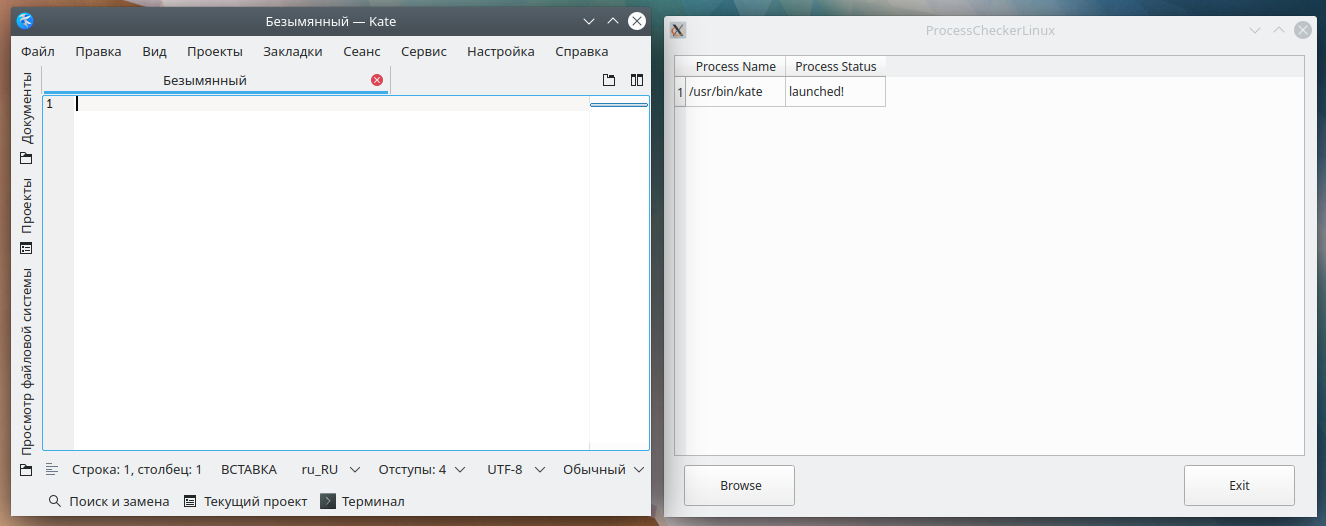


Рисунок 9 – Вывод сведений о состоянии процесса в Linux

* Следующим этапом было разбиение функционала данного приложения на две части: клиентскую, которая отображает полученную информацию на экране (Рисунок 10), и серверную, которая непосредственно получает информацию о процессах (Рисунок 11, Рисунок 12).

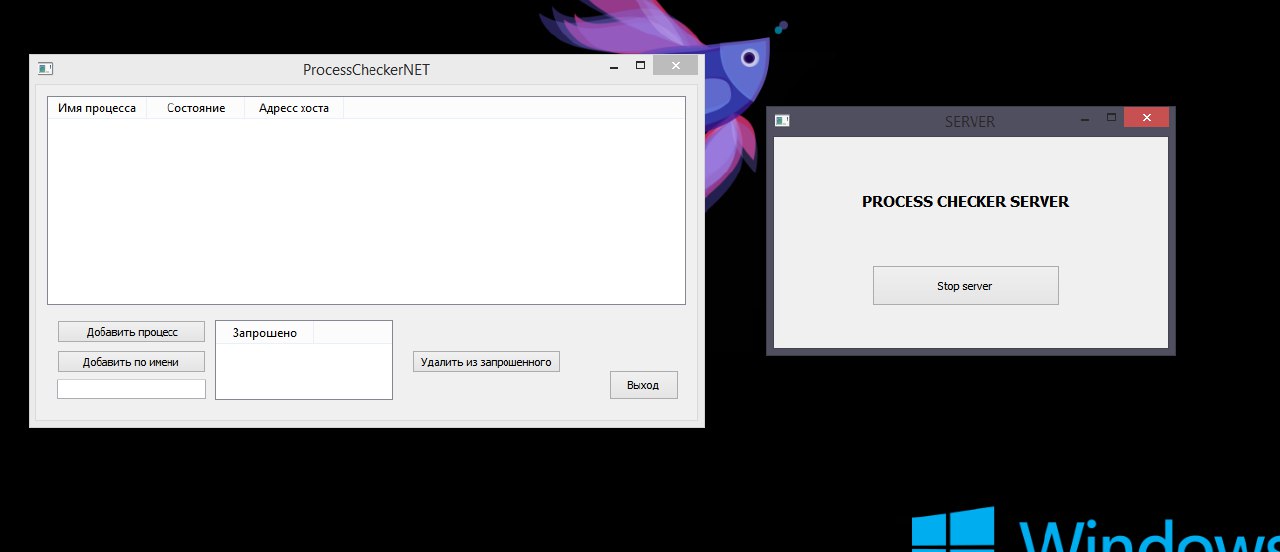


Рисунок 10 – Клиентская часть приложения

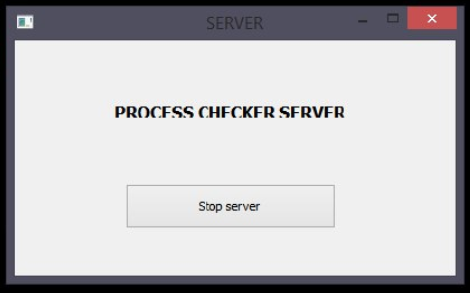


Рисунок 11 – Интерфейс сервера под Windows

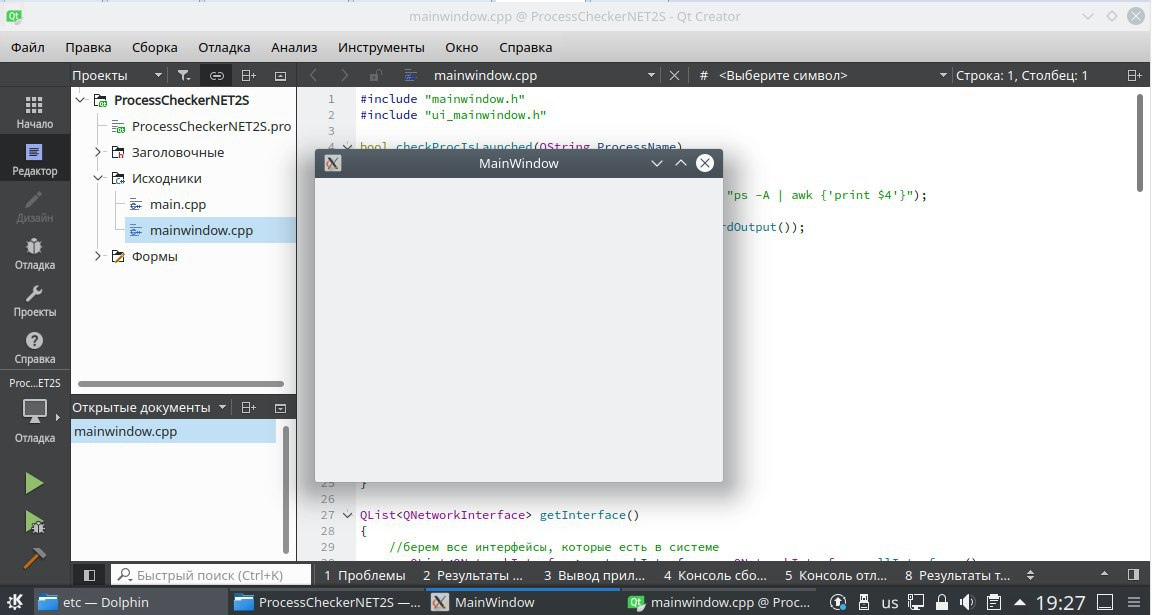


Рисунок 12 – Интерфейс сервера под Linux

* Далее был реализован функционал для опроса состояния удаленных узлов, работающий по принципу ожидания ответа от сервера. Сведения о доступности узла также отображаются в клиентской части (Рисунок 13). Все сведения об опрашиваемых узлах загружаются из заранее подготовленного текстового файла (Рисунок 14).

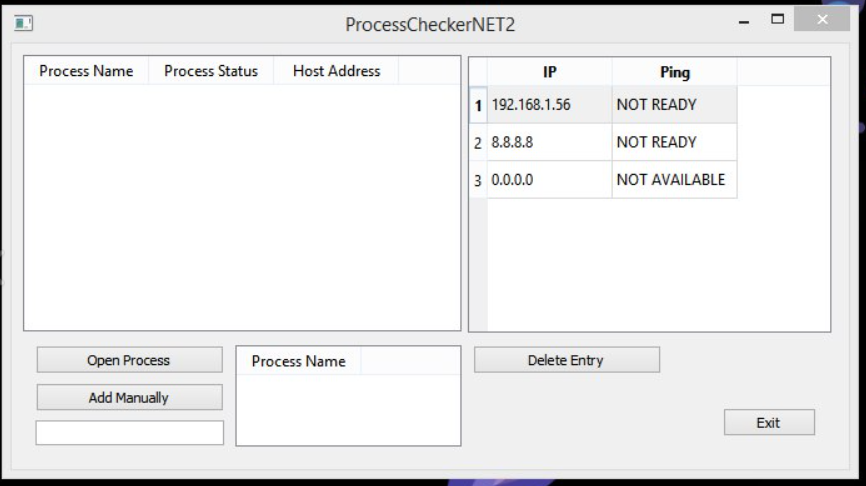


Рисунок 13 – Вывод сведений о доступности узла

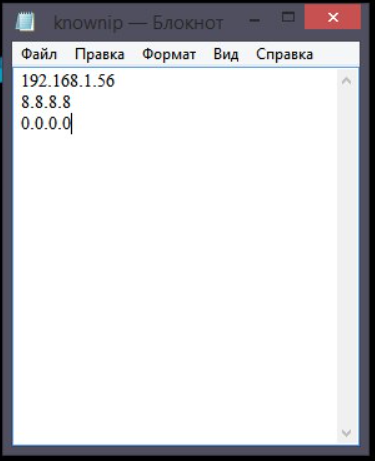


Рисунок 14 – Файл с заранее подготовленными IP - адресами

* Потом было реализовано получение сведений о загрузке отдельных аппаратных ресурсов компьютера, которые в процентном эквиваленте отображаются у клиента (Рисунок 15).

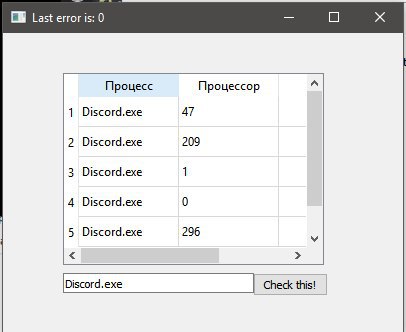


Рисунок 15 – Вывод сведений о состоянии загрузки процессора отдельными программами.

* 1. Доступ к результатам

Исходные коды всех модулей работающего приложения были загружены в репозиторий GitHub по адресу: https://github.com/Chao0001/NIR2020.

1. Заключение

В ходе выполнения данного индивидуального проекта была разработаны программные модули, выполняющие три задачи:

* Определение и вывод в табличной форме текущего состояния процессов, запущенных на текущем компьютере и удаленных компьютерах.
* Определение и вывод в табличной форме сведений о состоянии удаленных узлов.
* Определение и вывод в табличной форме сведений о загруженности аппаратных компонентов узла запущенными процессами.

Для реализации графических интерфейсов и обмена данными по сетям мною были освоены соответствующие библиотеки и классы фреймворка Qt.

Для реализации функций получения сведений о нагрузке на узел, даваемой отдельными процессами в среде операционной системы Windows мной были освоены соответствующие средства WinAPI и WMI.

Для реализации функций получения сведений о нагрузке на узел, даваемой отдельными процессами в среде операционных систем на базе ядра Linux мной были освоены соответствующие системные вызовы и особенности архитектуры ядра Linux, такие как файловая система procfs, которая позволяет хранить сведения о процессах в виде специальных файлов.

Данное задание не было реализовано в полном объеме. В дальнейшем, планируется интеграция данных модулей в единую систему удаленного мониторинга, которая позволяет централизованно получать сведения как об отдельных компонентах удаленных узлов, так и обо всех узлах в целом.

Список использованных источников

1). https://evileg.com/ru/post/373/

2). https://doc.qt.io/qt-5/qprocess.html

3). https://inlnk.ru/MjdMn