Министерство образования и науки Челябинской области

государственное бюджетное профессиональное

образовательное учреждение

«Златоустовский индустриальный колледж им. П.П. Аносова»

**ЗАЩИТА**

Руководитель УП ПМ.03

Преподаватель ГБОУ ЗлатИК

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.В.Майер

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

По учебной практике

Специальность: 09.02.07 «Информационные системы и программирование»

ПМ.03 «Ревьюирирование программных продуктов»

Выполнил:

Студент группы\_\_ИС-32\_\_

\_\_\_\_\_\_\_\_\_\_\_\_И.А.Федосов

\_\_\_\_\_\_\_\_\_\_\_дата

2020-2021 уч.г.

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc88897826)

[1. Список терминов и определений 7](#_Toc88897827)

[2.Функциональные требования 9](#_Toc88897828)

[3. Измерение производительности приложения посредством анализа использования ЦП 10](#_Toc88897829)

[4. Установка программного обеспечения для обратного проектирования 18](#_Toc88897830)

[5. Дизассемблирование 22](#_Toc88897831)

[6. Обратное проектирование 23](#_Toc88897832)

[ЗАКЛЮЧЕНИЕ 26](#_Toc88897833)

[ЛИТЕРАТУРА 2](#_Toc88897834)7

[Приложение 1 28](#_Toc88897835)

План прохождения учебной практики

по ПМ.03 «Ревьюирирование программных продуктов»

Студент Федосов Артём Игоревич

Группа ИС-32

Специальность 09.02.07 «Информационные системы и программирование»

Количество часов 36

Руководитель практики Ю.В.Майер

Сроки проведения практики 22.11.-27.112021

Таблица 1 – План прохождения практики

|  |  |  |
| --- | --- | --- |
| Этапы практики | Дата | Освоен  (да/нет) |
| Измерение производительности приложения посредством анализа использования ЦП | 22.11 |  |
| Установка ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования | 23.12 |  |
| Дизассемблирование. | 24.12 |  |
| Обратное проектирование. | 25.12 |  |
| Разработать сопроводительную документацию | 26.12 |  |
| Защита практики | 27.12 |  |

**Отзыв руководителя**

Студент Федосов Артём Игоревич

Группа ИС-32

Специальность 09.02.07 «Информационные системы и программирование»

Количество часов 36

Руководитель практики Ю.В.Майер

Сроки проведения практики 22.11.-27.112021

Место работы ГБПОУ «Златоустовский индустриальный колледж им. П.П. Аносова»

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Показатели и критерии оценивания** | **Баллы** | **Фактический балл** |
| **1** | **Структура отчета по практике** | | |
|  | Структура полностью соответствует заданию | 3 |  |
| Структура имеет несущественное несоответствие заданию | 2 |  |
| Структура существенно не соответствует заданию | 1 |  |
| **2** | **Соответствие содержания рабочей теме,цели,задачам** |  |  |
|  | Полное соответствие | 3 |  |
| Частичное соответствие | 2 |  |
| Низкая степень соответствия | 1 |  |
| **3** | **Полнота раскрытия темы** | | |
|  | Тема раскрыта полностью, приведены необходимые расчеты, пояснения, аргументы, сделаны выводы | 3 |  |
| Тема раскрыта полностью, однако приведены не все необходимые расчеты, пояснения и аргументы | 2 |  |
| Тема раскрыта частично, нет необходимых расчетов, пояснений, аргументов, не сделаны выводы | 1 |  |
| **4** | **Логика изложения материала** | | |
|  | Все структурные элементы организованы в систему, прослеживается логика в раскрытии темы | 3 |  |
| Все структурные элементы организованы в систему, логика в раскрытии темы частично нарушена | 2 |  |
| Все структурные элементы организованы в систему, но нет логики в раскрытии темы | 1 |  |
| **5** | **Соблюдение требований ГОСТ к оформлению ПЗ** | | |
|  | Требования ГОСТ соблюдены полностью | 3 |  |
| Имеются незначительные отклонения от ГОСТ | 2 |  |
| Есть существенные нарушения требований ГОСТ | 1 |  |
| **6** | **Практическая часть** | | |
|  | Выполнена в соответствии с требованиями, без отклонений от нормативов | 3 |  |
| Имеется несущественное отклонение от нормативов | 2 |  |
| Имеется существенное отклонение от нормативов | 1 |  |
| ИТОГО | | |  |

Критерии оценивания:

18-15 б. – «отлично»;

11-14 б. – «хорошо»;

8-10 б. – «удовлетворительно»;

Меньше 8 б. работа не оценивается.

Подпись руководителя:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.В.Майер

\_\_\_\_\_\_\_\_\_\_\_\_\_дата

# ВВЕДЕНИЕ

Процессы разработки, приобретения и внедрения сложных систем, к которым относятся в частности программные комплексы, должны находится под жестким управленческим контролем. В настоящее время практически во всех организациях обеспечивается контроль важнейших характеристик, связанных с производством и использованием программных продуктов, таких как время, финансовые средства, ресурсы и т.п. Однако в большинстве случаев вне пределов сферы контроля оказывается наиболее важная характеристика программных продуктов, ради которой, собственно и осуществляются затраты времени, финансовых средств и ресурсов – это качество продукта, поскольку «невозможно контролировать то, что нельзя измерить» (“You cannot control what you cannot measure”).

Дизассемблирование **–** преобразованиепрограммы на машинном языке к ее ассемблерному представлению. Декомпиляция – получение кода языка высокого уровня из программы на машинном языке или ассемблере.

Под *анализом потоков данных* понимают совокупность задач, нацеленных на выяснение некоторых глобальных свойств программы, то есть извлечение информации о поведении тех или иных конструкций в некотором контексте.

Основным результатом деятельности группы разработчиков являются не диаграммы, а программное обеспечение, поэтому модели и основанные на них реализации должны соответствовать друг другу с минимальными затратами по поддержанию синхронизации между ними. Чаще всего разработанные модели преобразуются в программный код. Хотя UML не определяет конкретного способа отображения на какой-либо объектно-ориентированный язык, он проектировался с учетом этого требования. В наибольшей степени это относится к диаграммам классов, содержание которых без труда отображается на такие известные объектно-ориентированные языки программирования, как Java, C++, ObjectPascal, Visual Basic и др.

*Прямым проектированием*(Forward engineering) называется процесс преобразования модели в код путем отображения на некоторый язык реализации.

*Обратным проектированием*(Reverse engineering) называется процесс преобразования в модель кода, записанного на каком-либо языке программирования.

Цель учебной практики: Научиться выполнять прямое и обратное проектирование ПО, овладеть навыками приостановки и возобновления работы с рабочими задачами иинспекцией кода в Visual Studio, совместно работать над проектом в системе контроля версий GIT.

В ходе учебной практики для достижения цели, ставятся задачи:

* измерить производительность приложения посредством анализа использования ЦП,
* установить ПО (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,
* выполнить дизассемблирование,
* выполнить обратное проектирование.

# 1. Список терминов и определений

**Репозиторий** (от англ. repository – склад, хранилище; встречается также написание «репозитарий») – это профессиональный участник рынка ценных бумаг, осуществляющий ведение реестров договоров РЕПО и договоров с деривативами, заключенных на организованных торгах

**Дизассемблер** —это компьютерная программа, которая переводит машинный язык на язык ассемблера-операция, обратная операции ассемблера.

**Radare2** (также известный как r2)-это полный фреймворк для обратного проектирования и анализа двоичных файлов, состоящий из набора небольших утилит, которые могут использоваться вместе или независимо от командной строки.

**Точка останова** (англ. breakpoint) — это преднамеренное прерывание выполнения программы, при котором выполняется вызов отладчика (одновременно с этим программа сама может использовать точки останова для своих нужд).

**Компилятор** - это программа, которая выполняет преобразование текста программы в другое представление, обычно машинный код, без его запуска, статически.

**GCC** - это свободно доступный оптимизирующий компилятор для языков C, C++, Ada 95, а также Objective C. Его версии применяются для различных реализаций Unix (а также VMS, OS/2 и других систем PC), и позволяют генерировать код для множества процессоров.

**Программное обеспечение** (ПО) — программа или множество программ, используемых для управления компьютером (ISO/IEC)

**Linux** — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты.

**GitHub** — это платформа, хранящая различные Git-репозитории на своих многочисленных серверах. Также GitHub называют крупнейшим веб-сервисом для хостинга и совместной разработки IT-проектов.

**Обратная разработка** — исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы; например, чтобы обнаружить недокументированные возможности (в том числе программные закладки), сделать изменение или воспроизвести устройство, программу или иной объект с аналогичными функциями, но без прямого копирования.

**Iaito** – графический интерфейс для radare2, написанного на С++ с использованием графического тулкита Qt 5.3

# 2.Функциональные требования

Для выполнения поставленных задач, мы использовали ПО Linux в нём мы устанавливали пакеты для работы с radare2, так же устанавливали приложение «Iaito» и к нему графический интерфейс.

Для выполнения задачи с измерение производительности приложений, нам нужна была программа Visual Studio.

Для того чтобы сохранить все наши сделанные задания, мы использовали приложение GitHub [(см. приложение 1).](#_Приложение_2)

Минимальные системные требования:

Процессор с тактовой частотой не ниже 1,8 ГГц. Рекомендуется использовать как минимум двухъядерный процессор.

2 ГБ ОЗУ; рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине).

Место на жестком диске: до 210 ГБ (минимум 800 МБ) свободного места в зависимости от установленных компонентов; обычно для установки требуется от 20 до 50 ГБ свободного места.

# 3. Измерение производительности приложения посредством анализа использования ЦП

1.Запускаем Visual Studio для работы с анализом программного кода, чтобы узнать нагрузку центрального процессора.

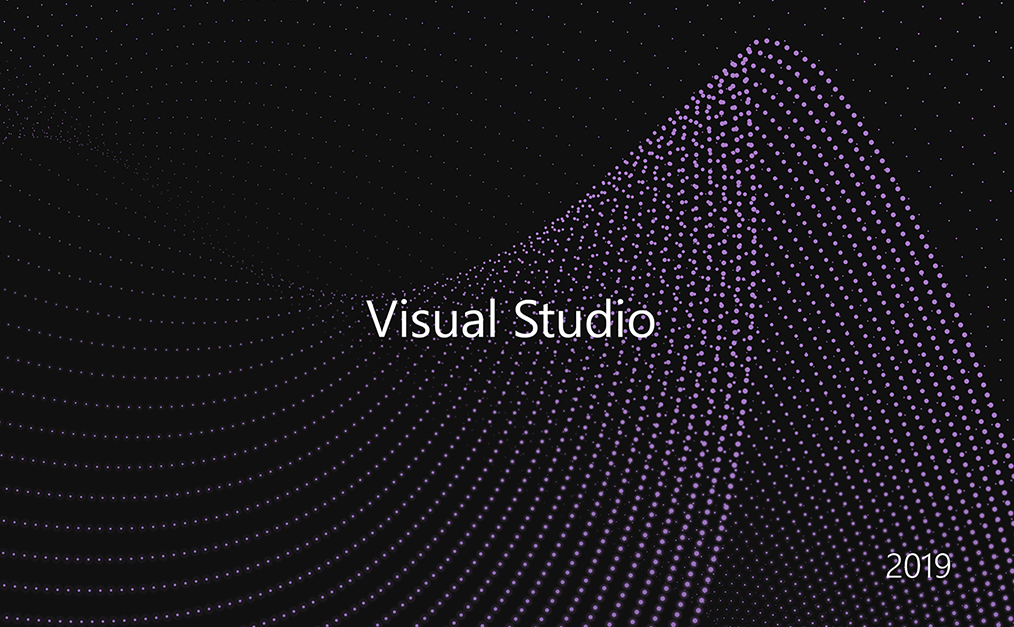


Рис.1 запуск Visual Studio

2. Открываем свой проект с программой, через клонирование репозитория.

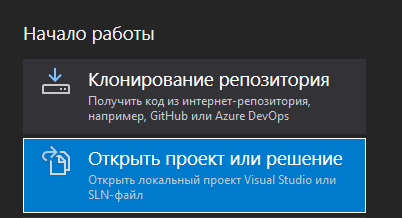


Рис.2 выбор проекта

3. Проект открылся



Рис.3 открытый проект

4. Установим точки остановы

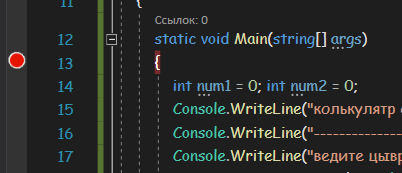


Рис.4 Первая точка остановы

5. Ставим вторую точку остановы

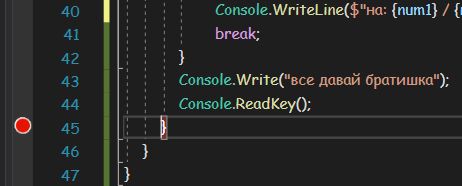


Рис.5 Вторая точка остановы

6. Открыли окно **Отладка>Окна>Показать средства диагностики**

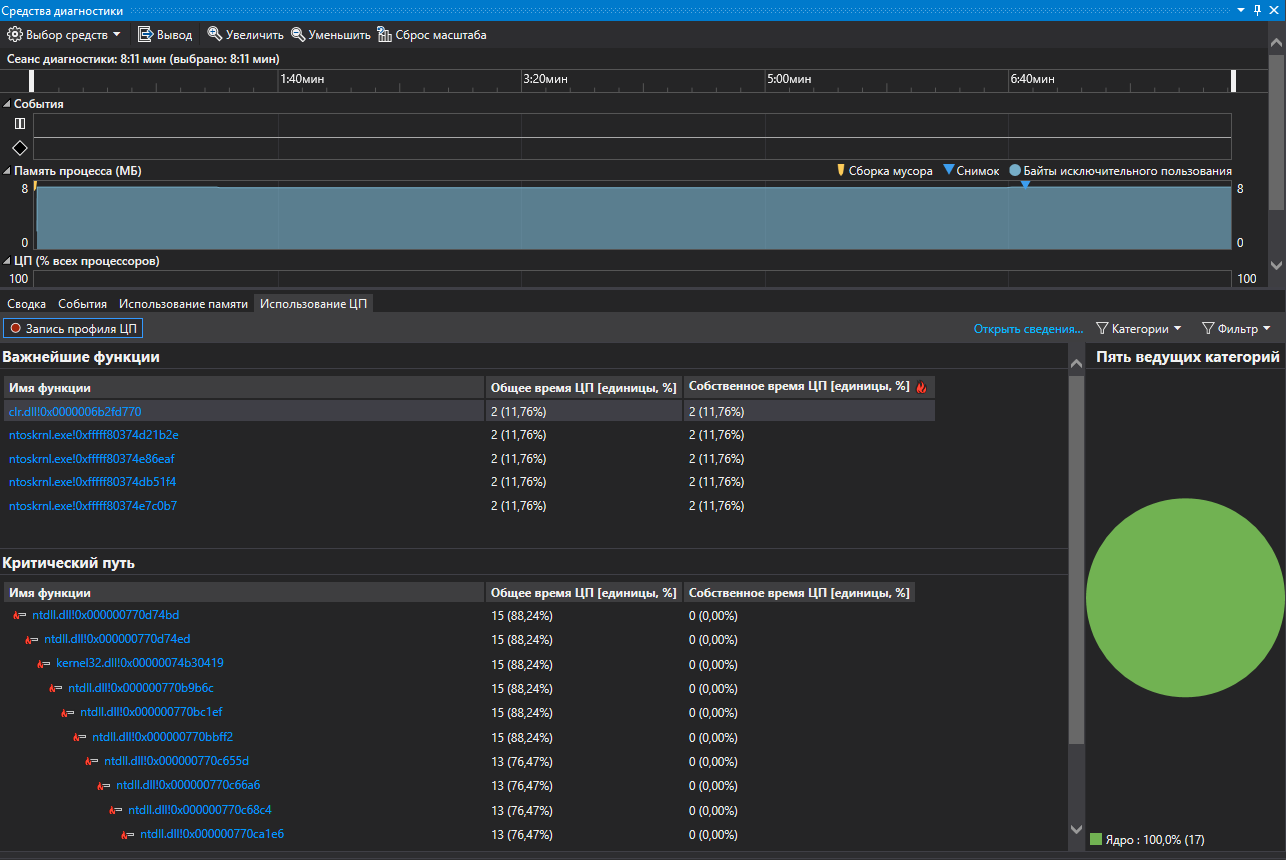


Рис.6 Окно средства диагностики

7. Вы можете выбрать, что следует просмотреть, [использование памяти](https://docs.microsoft.com/ru-ru/visualstudio/profiling/memory-usage?view=vs-2022) или **Загрузка ЦП** (либо оба средства), с помощью параметра **Выбор средств** на панели инструментов. В Visual Studio Enterprise также можно включить или отключить IntelliTrace, выбрав **Сервис** > **Параметры** > **IntelliTrace**.

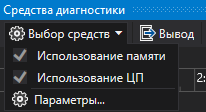


Рис.7 Параметры

8. **Отладка** > **Начать отладку**

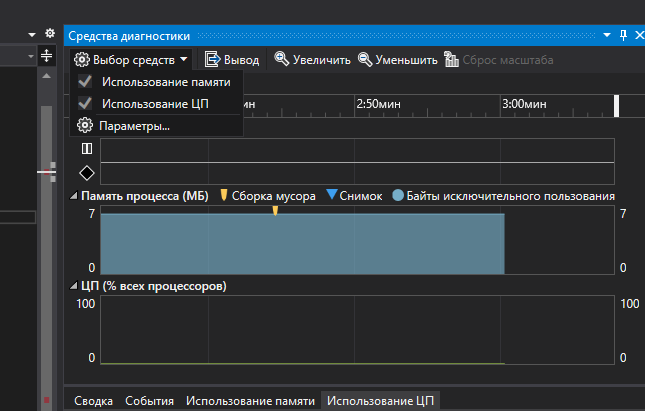


Рис.8 Отладка

9. Запустим сценарий, который вызывает срабатывание первой точки останова.

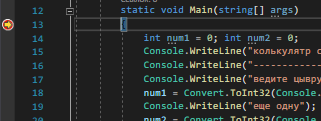


Рис.9 Срабатывание первой точки остановы

10. Приостановив отладчик, включим сбор данных о загрузке ЦП, а затем откроем вкладку **Загрузка ЦП**.



Рис.10 Кнопка включение сбора данных

11. Нажмите клавишу F5, чтобы запустить приложение до второй точки останова.

Теперь у вас есть данные о производительности приложения именно для той области кода, которая выполняется между двумя точками останова.

Профилировщик начинает подготавливать данные потока. Дождитесь завершения этой операции.

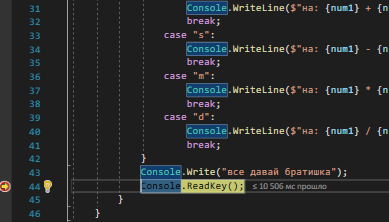


Рис.11 Срабатывания второй точки остановы

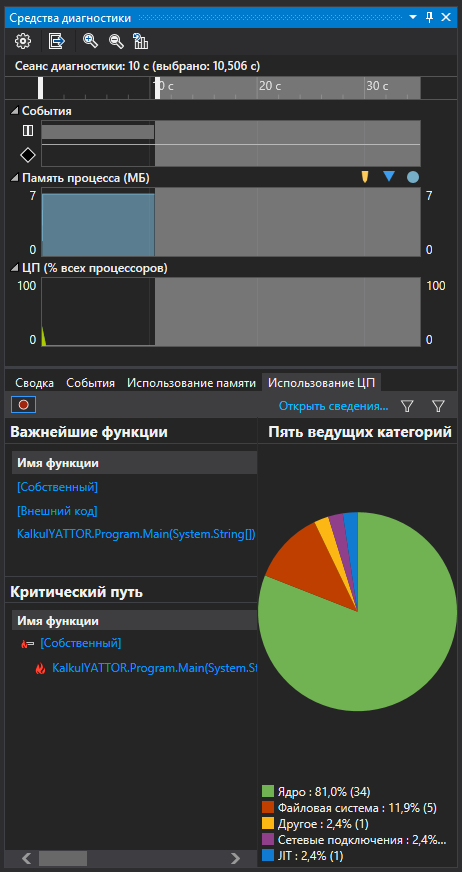


Рис.12 Данные о загрузке ЦП

12. Если вы хотите выбрать для анализа более конкретную область кода, выберите область на временной шкале ЦП (это должна быть область, в которой отображаются данные профилирования).



Рис.13 Временная шкала загрузки ЦП

**3.1. Ш**аг 2 Анализ данных о загрузке ЦП

1. В списке функций изучите функции, которые выполняют большую часть работы.

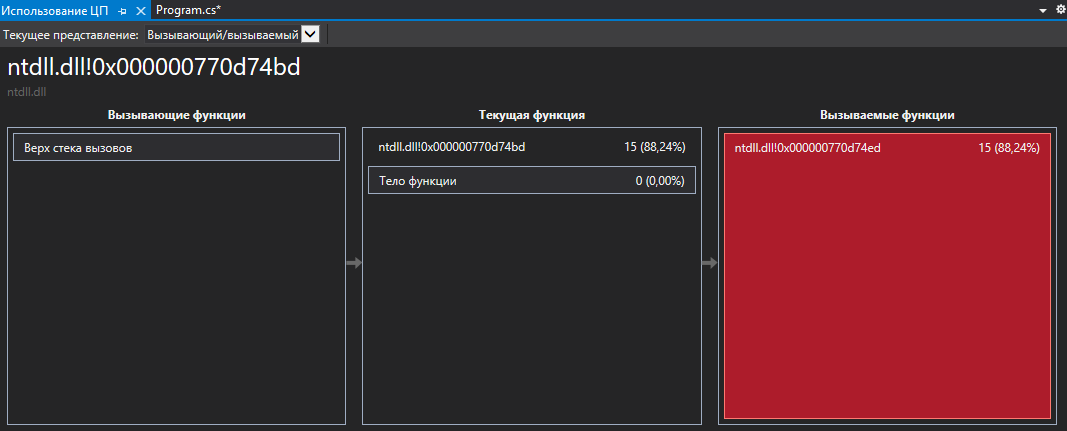
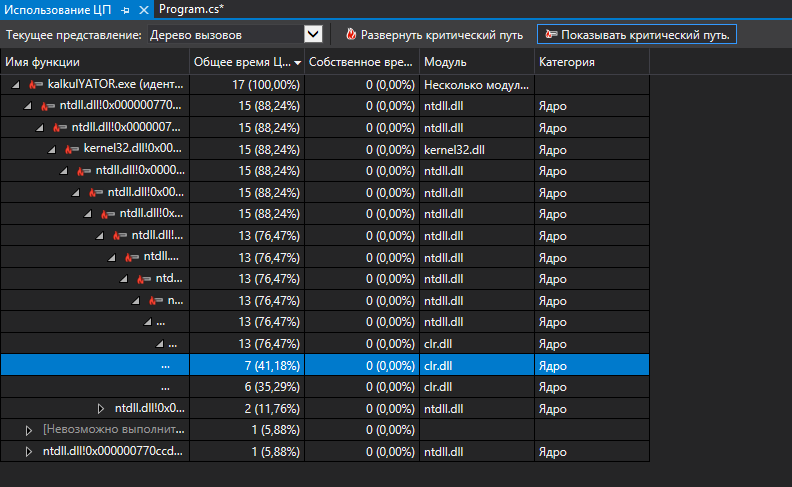


Рис.14 Вызываемые функции

2. Чтобы увидеть более обобщенное представление, показывающее порядок, в котором вызываются функции, выберите в раскрывающемся списке в верхней части панели пункт **Дерево вызовов**.

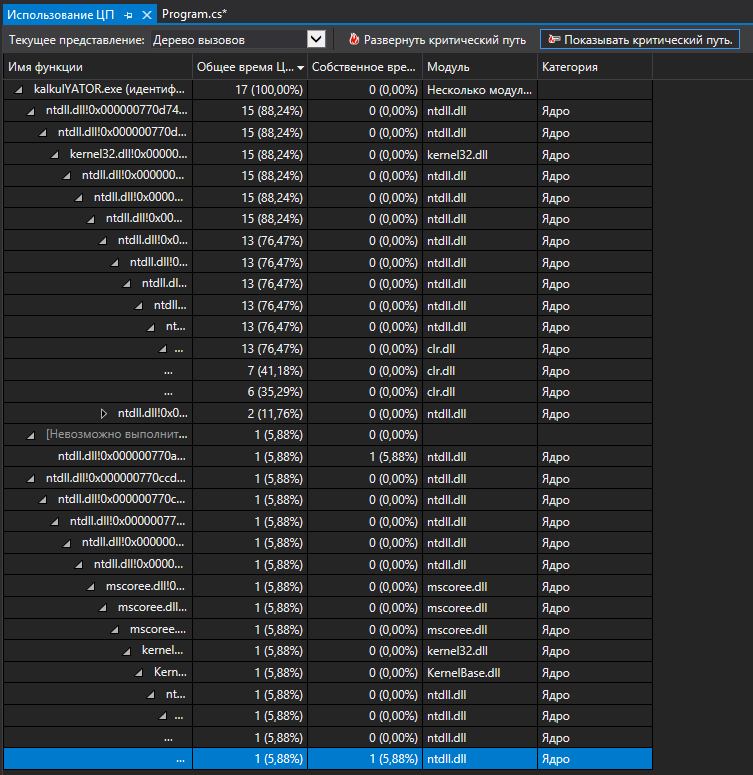
Каждая нумерованная область на рисунке соответствует определенному шагу в процедуре.

Рис.15 Дерево вызовов



Чтобы увидеть вызовы функций, которые используют самый высокий процент ЦП в представлении дерева вызовов, нажмите **Развернуть критический путь**.

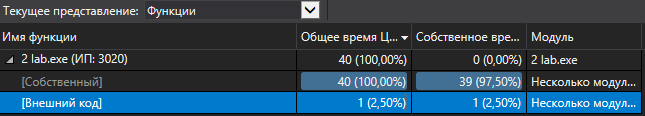
Рис.16 Критический путь



**3.2. Просмотр внешнего кода**

Внешний код — это функции в компонентах системы и платформы, которые исполняются вашим кодом. Внешний код включает функции, которые запускают и останавливают приложение, отрисовывают пользовательский интерфейс, управляют потоками и предоставляют приложению другие низкоуровневые службы. В большинстве случаев внешний код вас интересовать не будет, поэтому средство "Загрузка ЦП" собирает внешние функции пользовательского метода в один узел **[Внешний код]**

Если вы захотите посмотреть пути к вызовам внешнего кода, выберите **Показать внешний код** в списке **Представление фильтра** и выберите **Применить**.

Рис.17 Функции

4. **4.Установка ПО для обратного проектирования**

Создание виртуальной машины

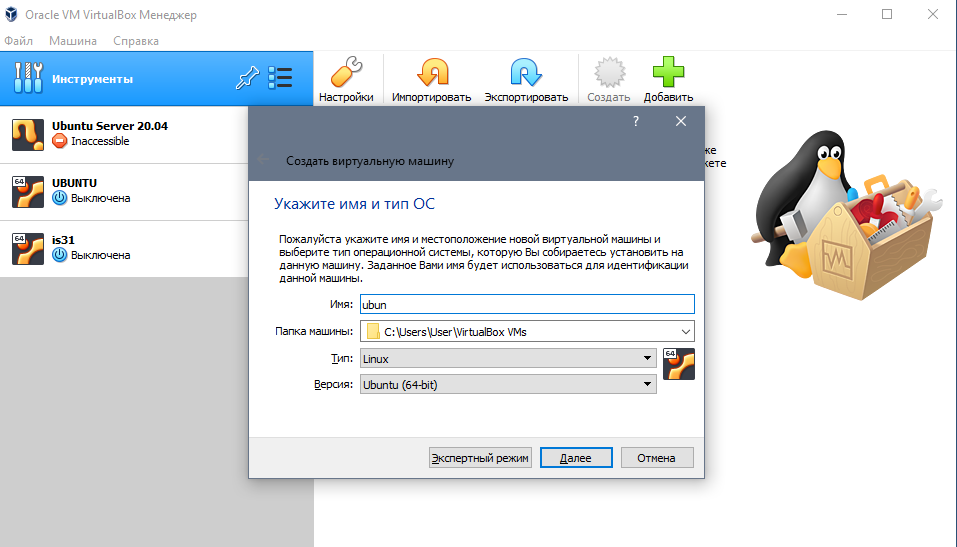


Рис.18 Виртуальная машина

Выбор языка

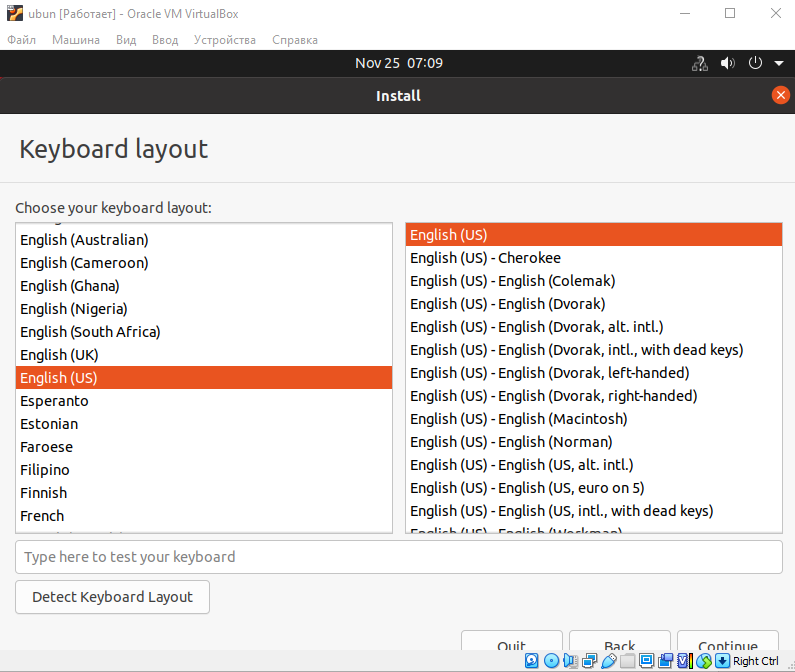


Рис.19 Язык

Выбор устанавливаемых приложений

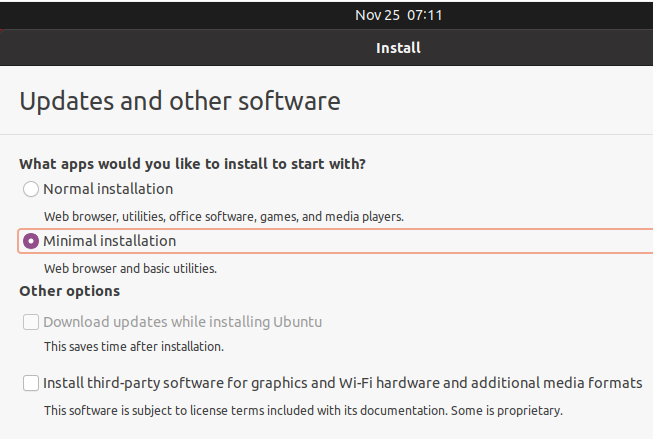


Рис.20 Установка приложенй

Выбор типа установки

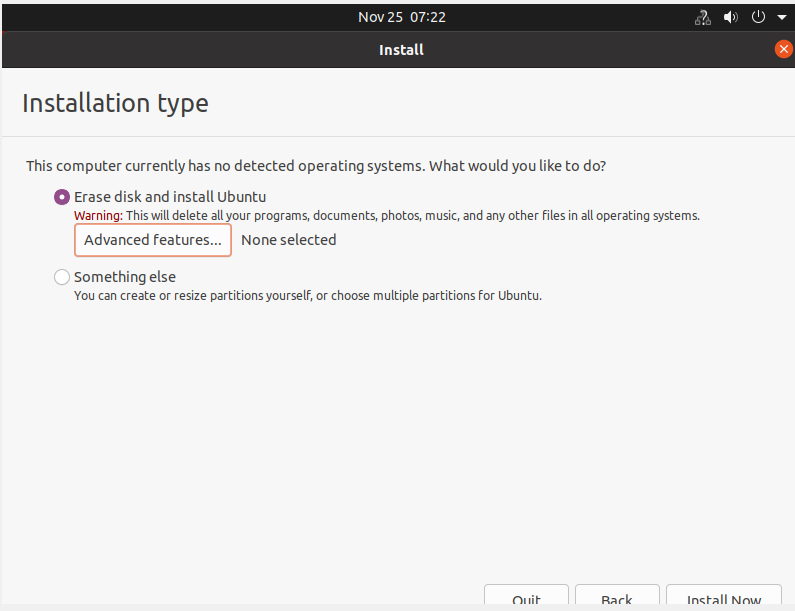


Рис.21 Тип установки

Выбор региона



Рис.22 Регион

Авторизация пользователя

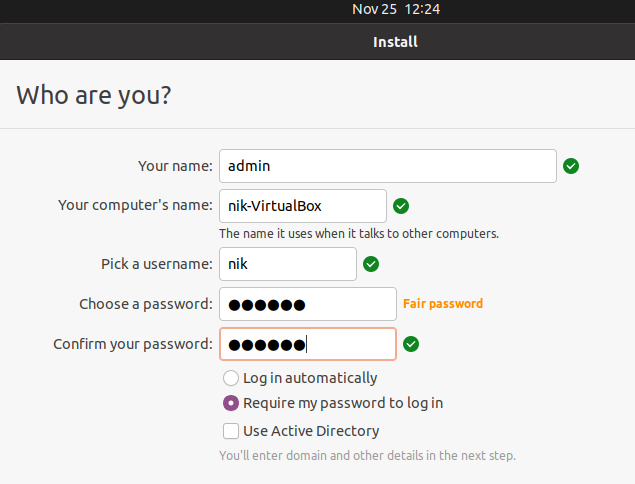


Рис.23 Авторизация

Вход в файл для прокси

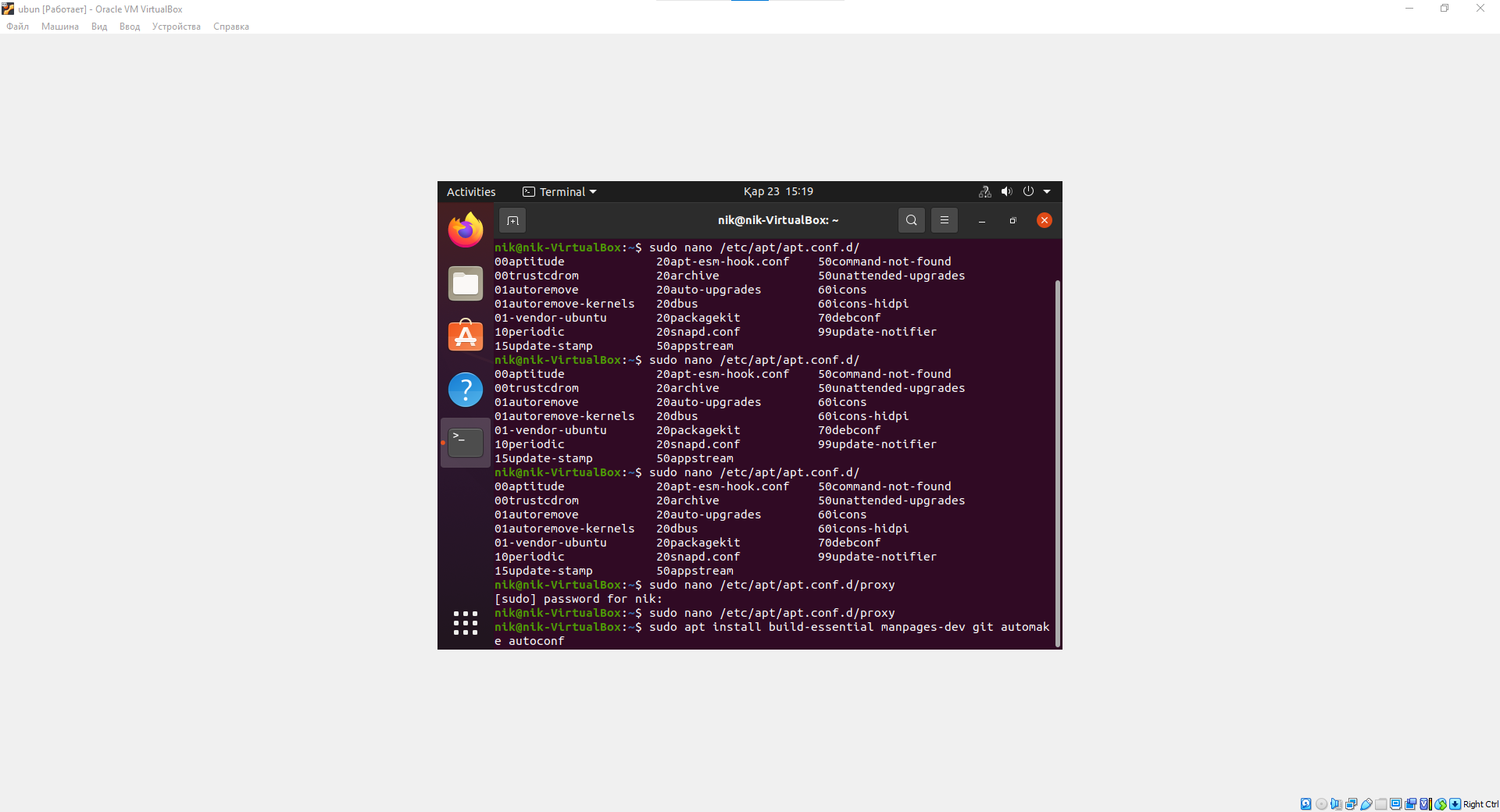


Рис.24 Файл прокси

Ввод в файл прокси колледжа

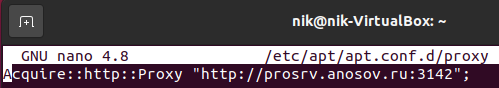


Рис.25 Прокси колледжа в файле

Установка компилятора gcc



Рис.26 Компилятор gcc

Установка пакетов для работы с radare2

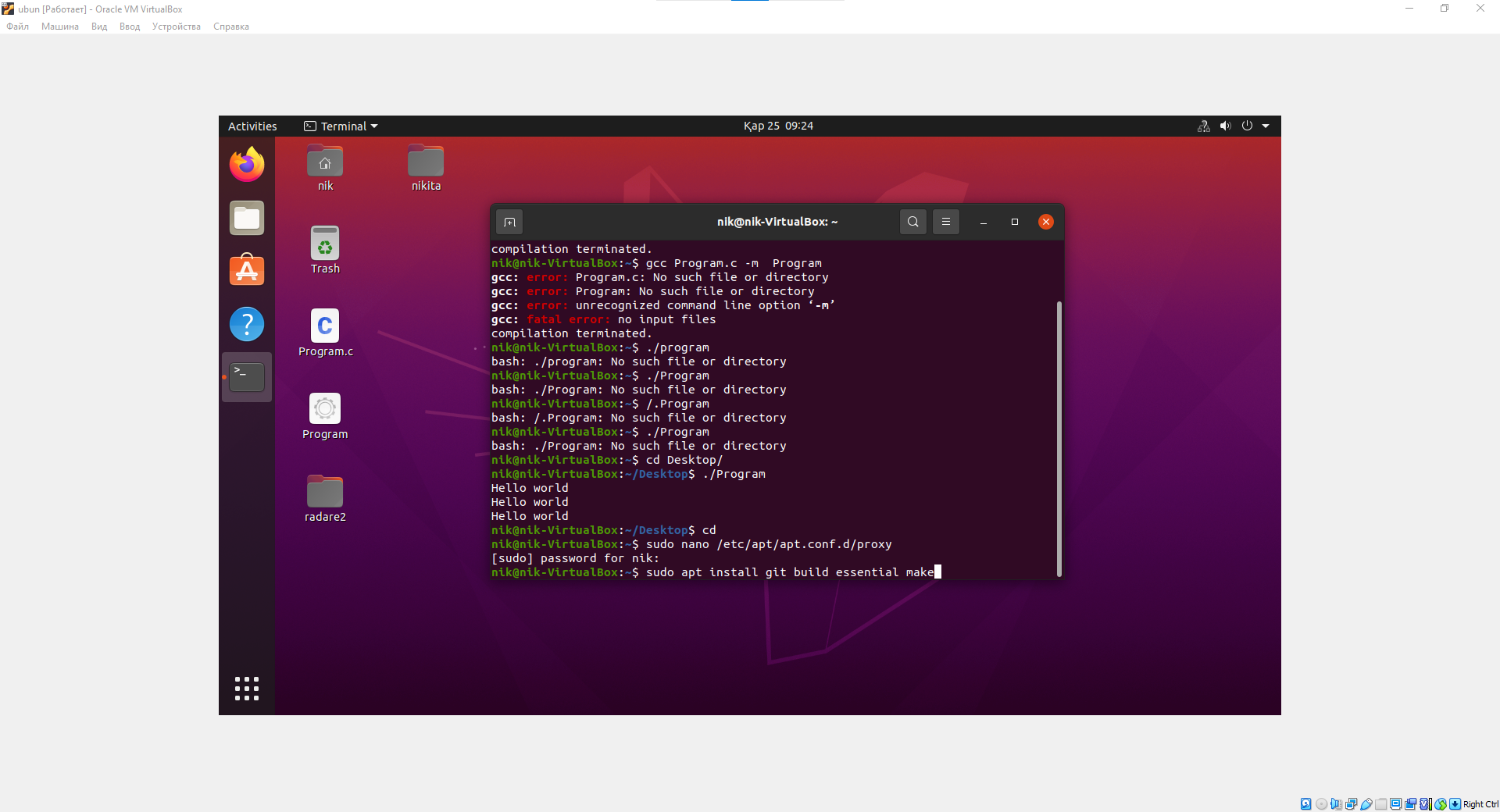


Рис.27 Пакеты radare2

Использование прокси колледжа

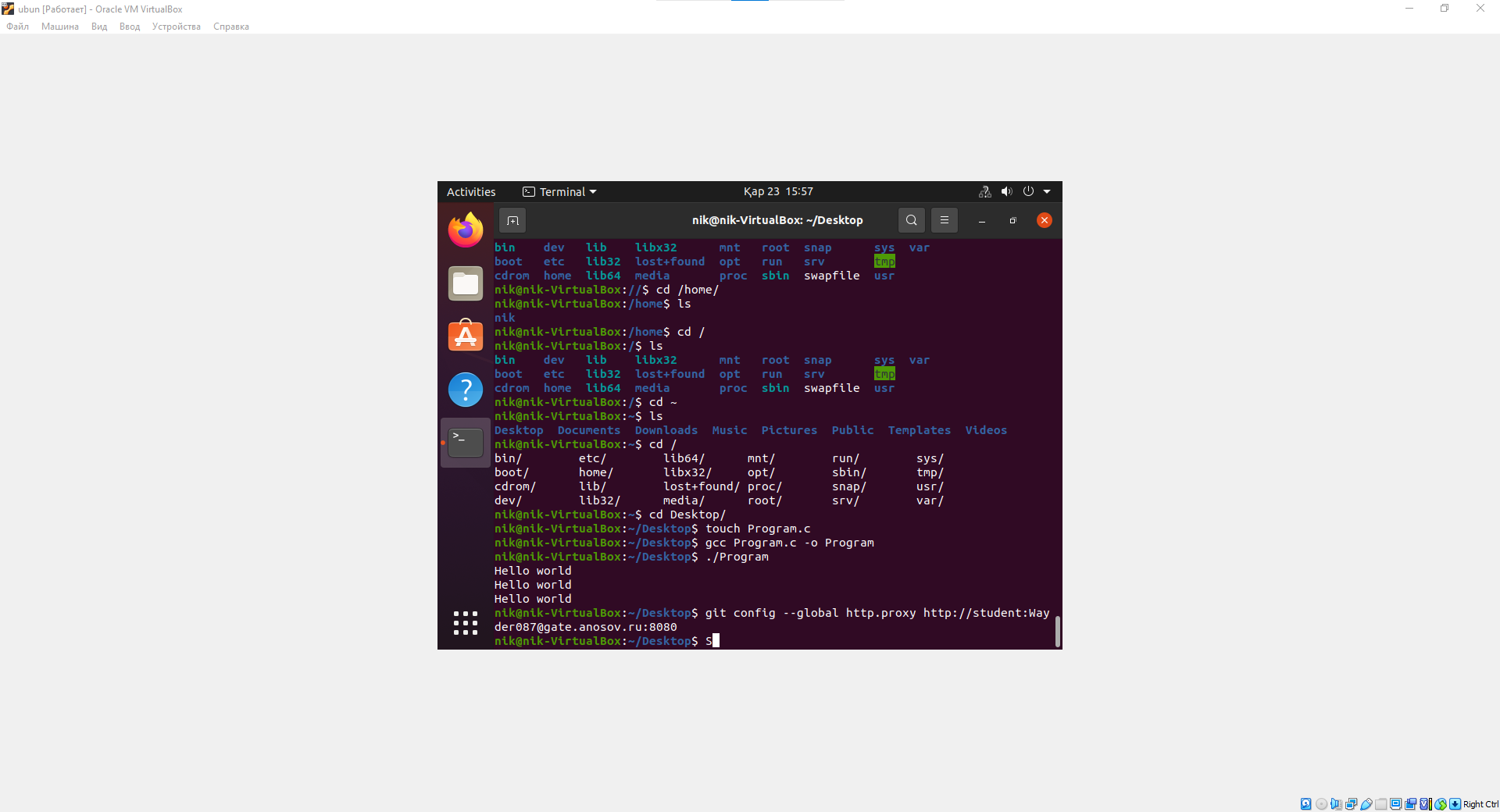


Рис.28 Прокси колледжа

Открытие файла для ввода прокси

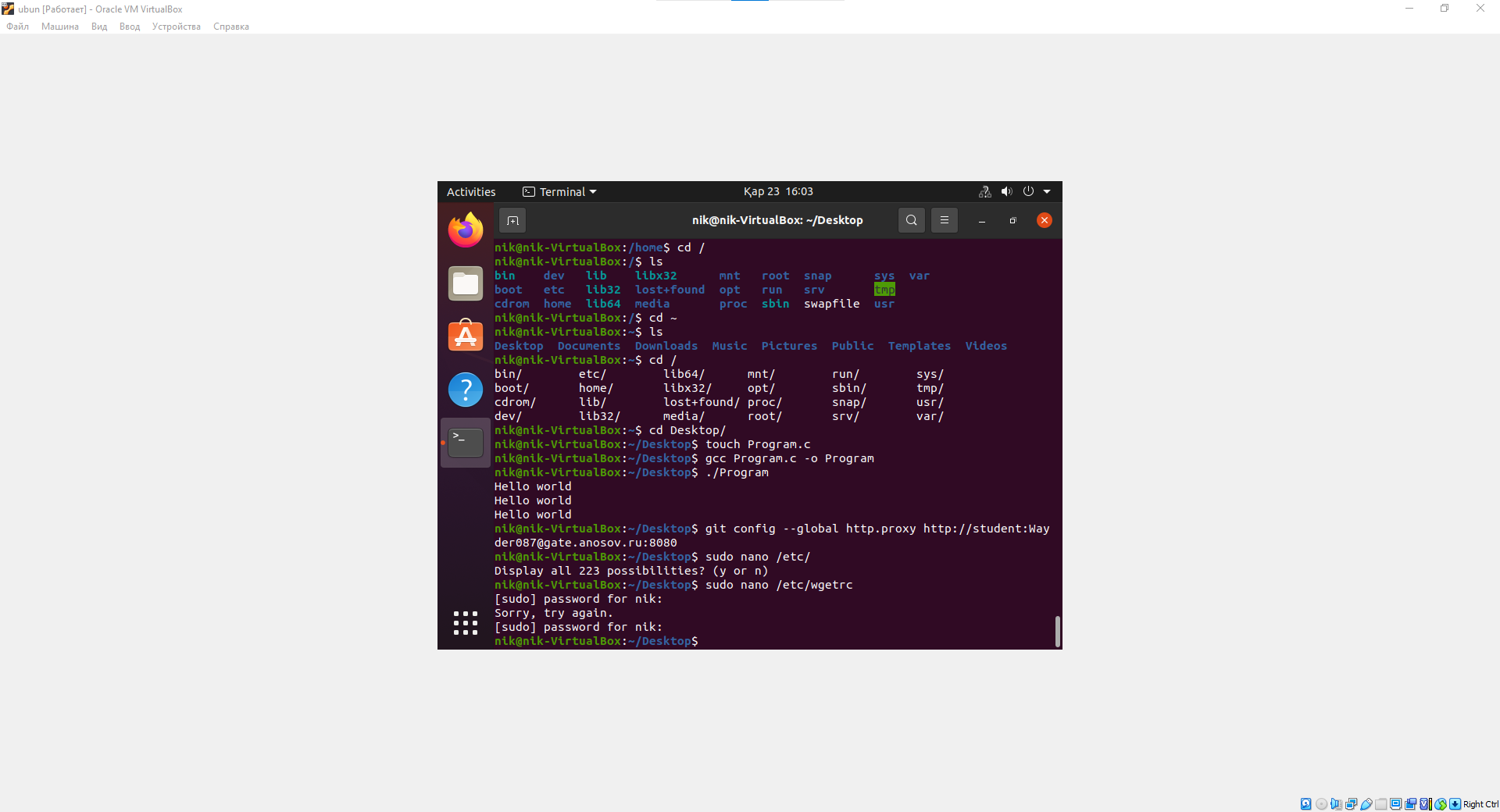


Рис.29 Открытие файла для прокси

Ввод прокси колледжа для http, https и ftp прокси

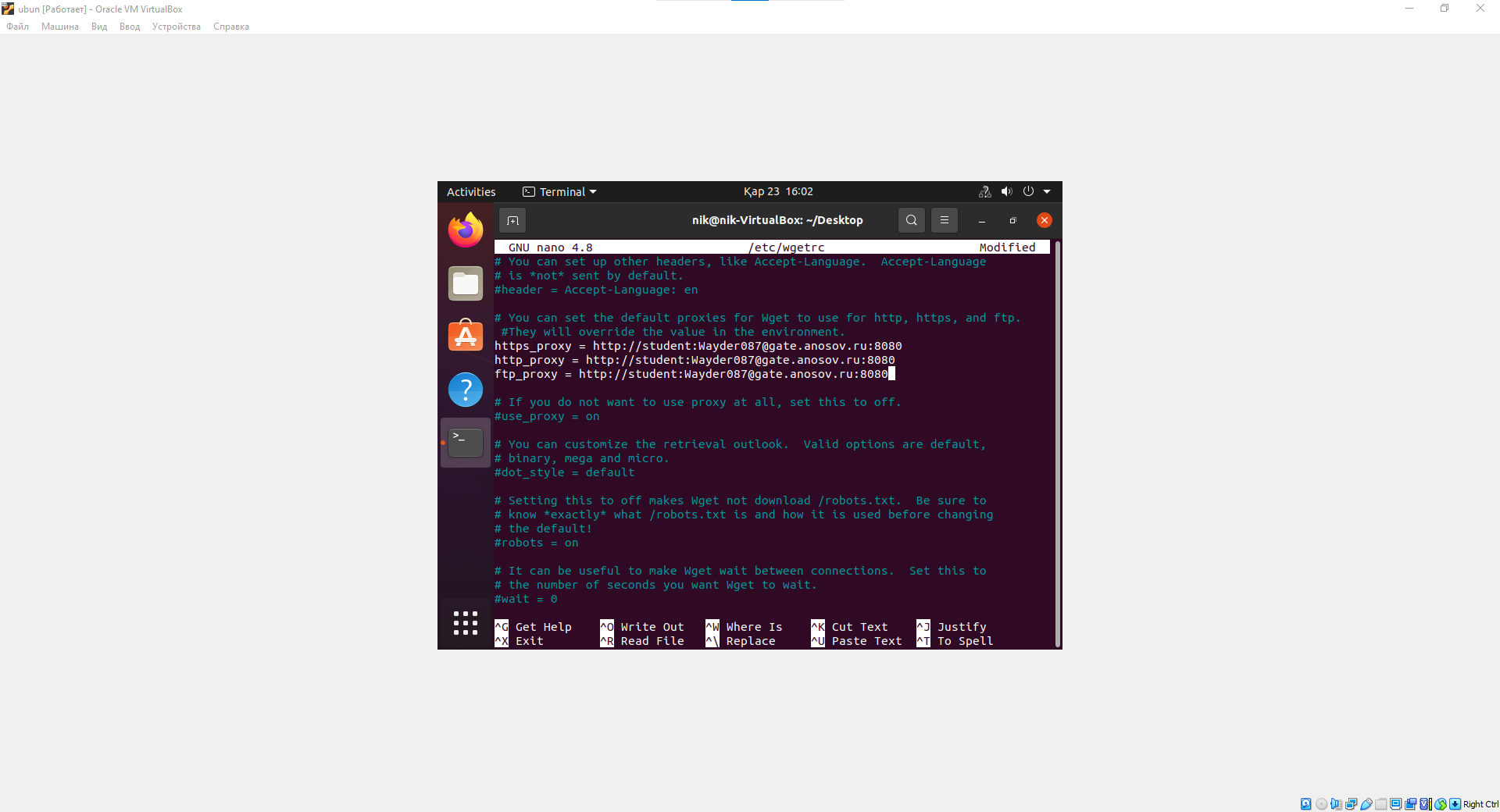


Рис.30 Файл с измененными прокси

Клонированние Radare2 с репозитория Git

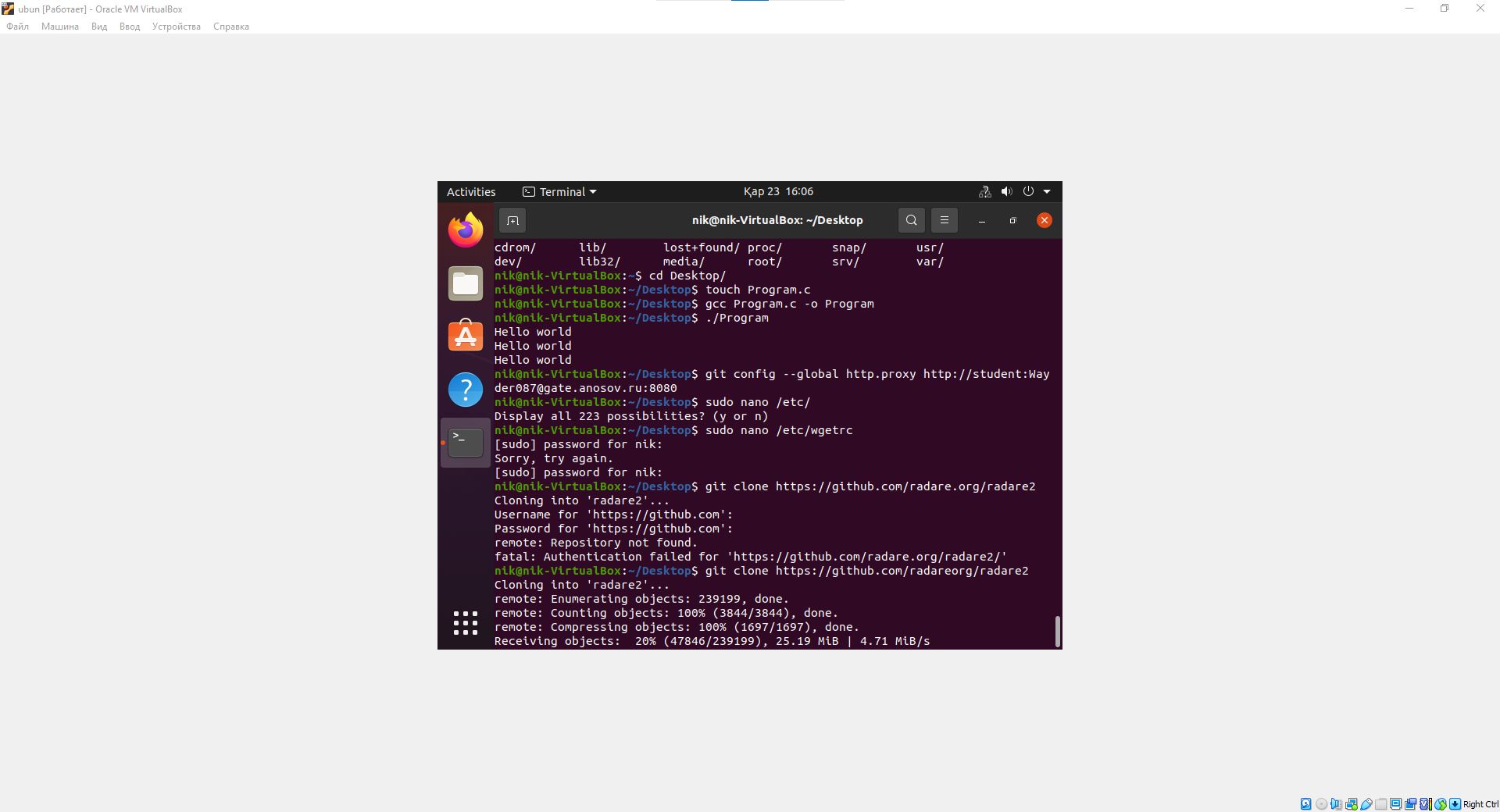


Рис.31 Скачивание radare2

Установка radare2 на рабочий стол

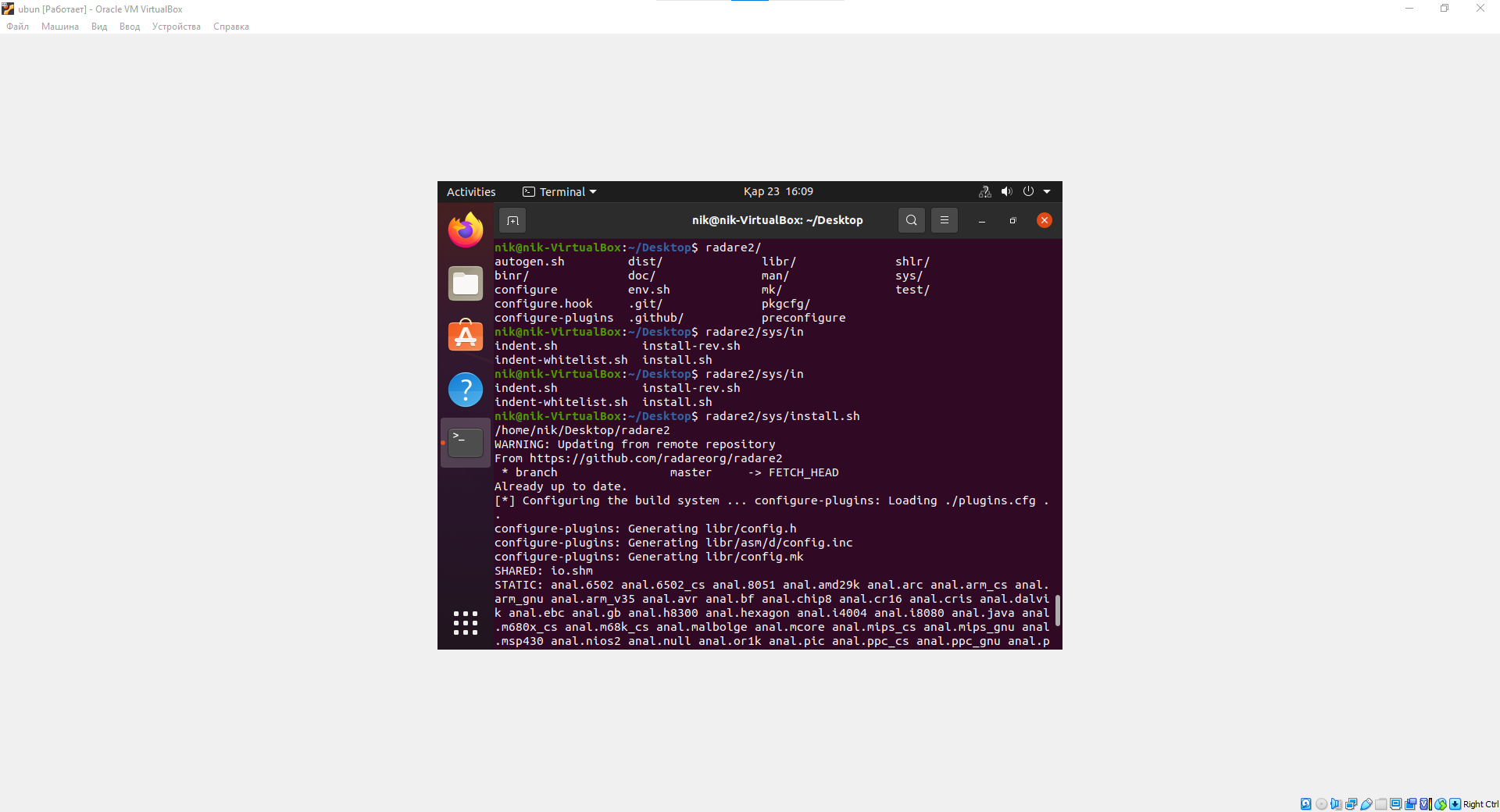


Рис.32 Установка radare2

Установка компонентов для графической оболочки iaito



Рис.33 Установка оболочки iaito

Переход в папку Radare2 и открытие файла конфигурации

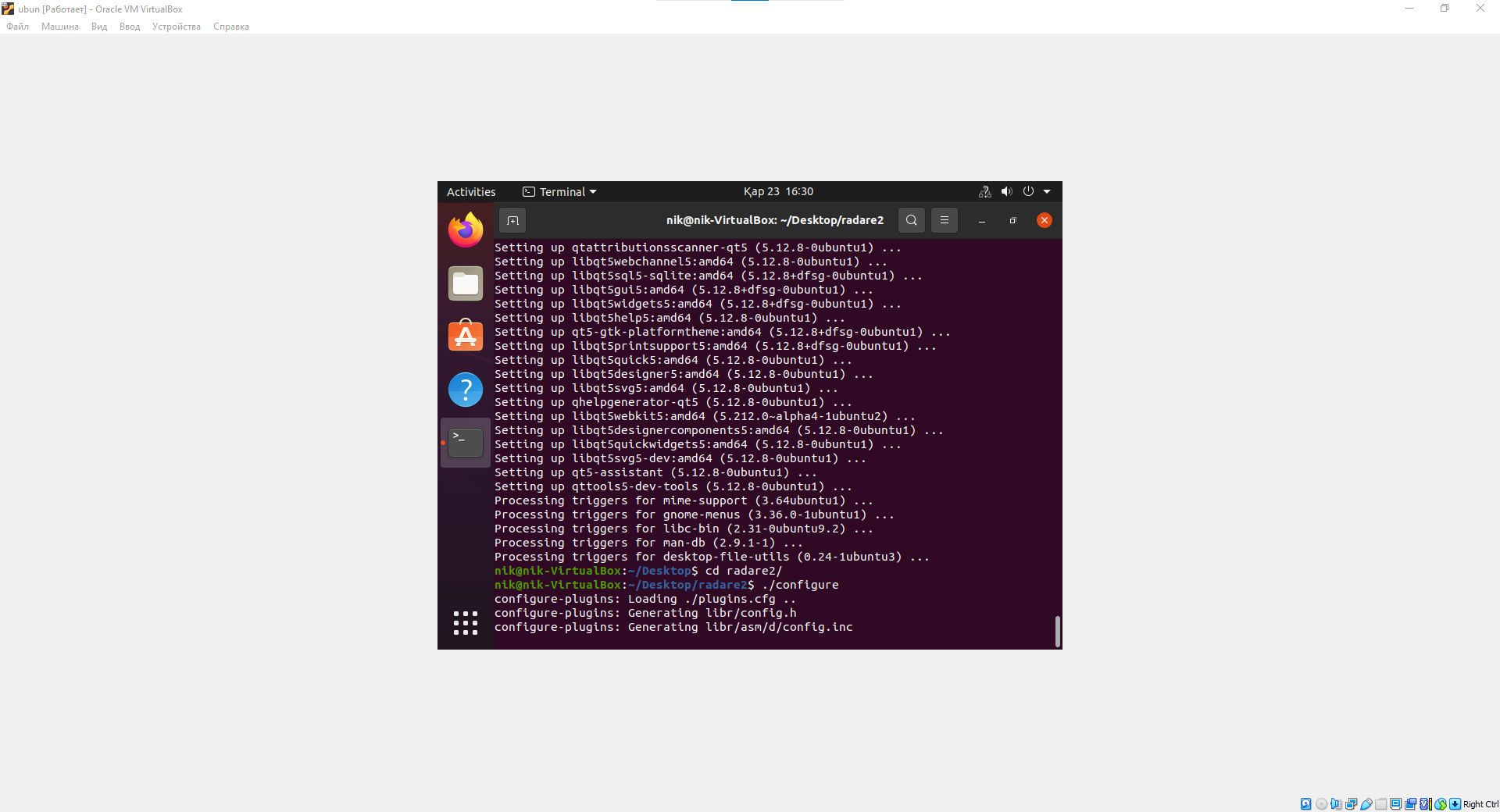


Рис.34 Конфигурация radare2

Распаковка radare2



Рис.35 Распаковка radare2

Установка графического интерфейса Iaito

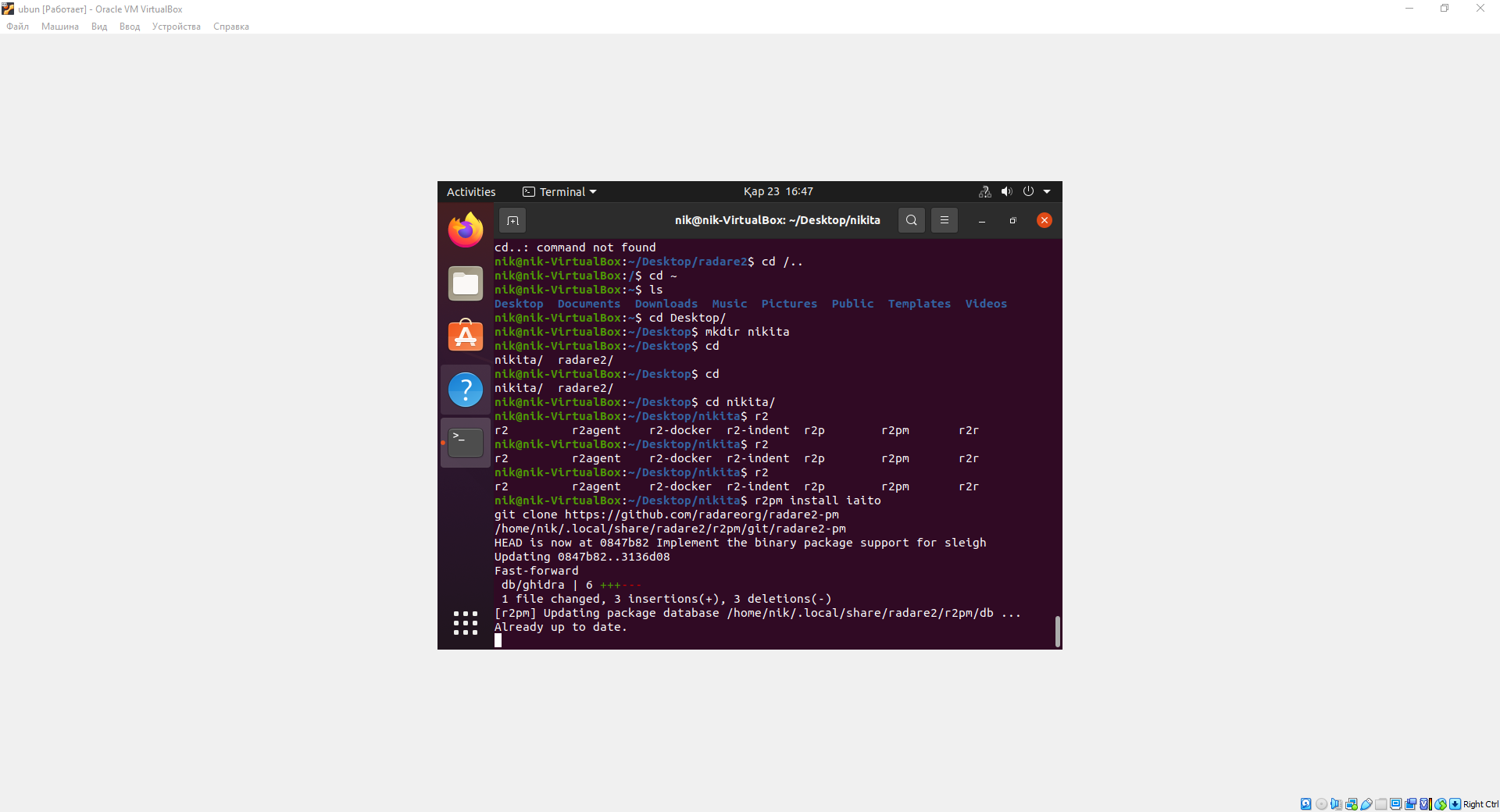


Рис.36 Графический интерфейс Iaito

Установка декомпилятора r2ghidra



Рис.37Декомпилятор r2ghidra

# 5.Дизассемблирование

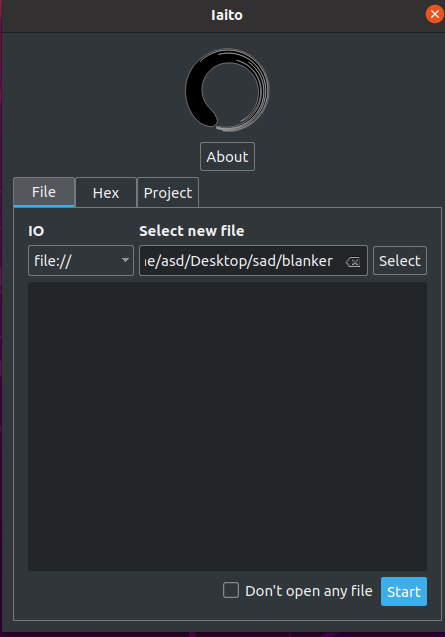


Рис.38 Вход в Iaito

Запущен Iaito и произведен переход во вкладку main

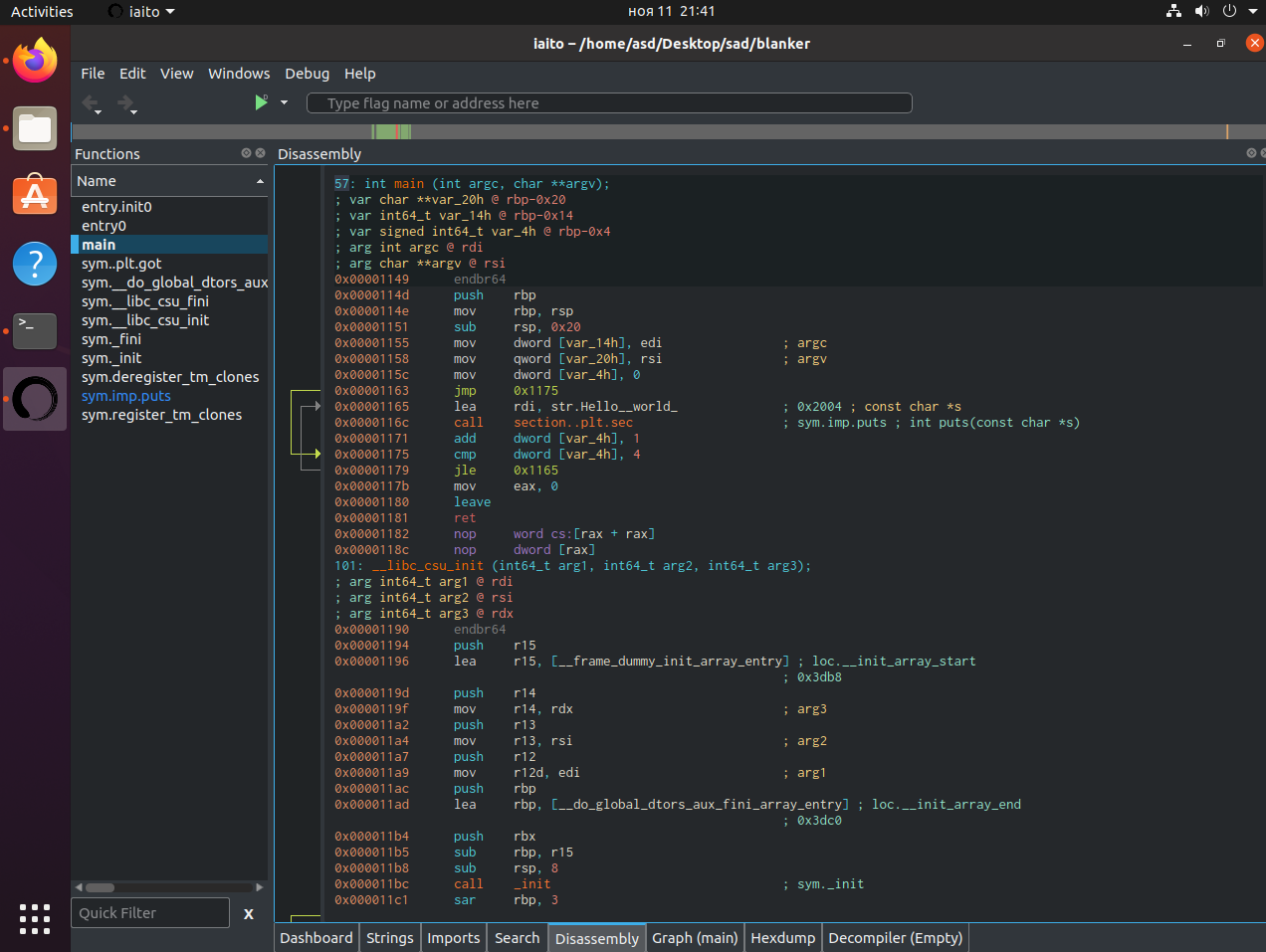


Рис.39 Iaito

Выбор элемента для изменения и его изменение(выделено)

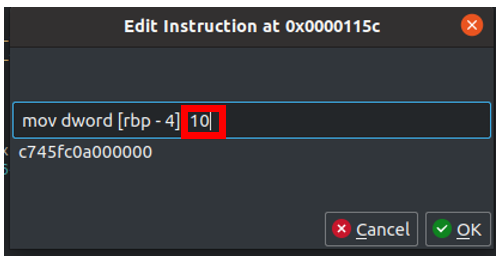


Рис.40 Изменение в коде

Проверка изменений

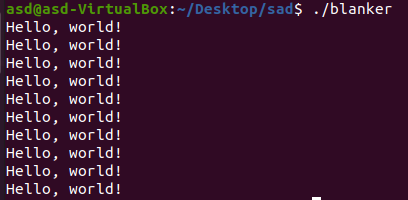


Рис.41 Результат изменений

# 6.Обратное проектирование

Создание файла с кодом для компиляции на рабочем столе



Рис.42 Файл для компиляции

Файл для компиляции на рабочем столе

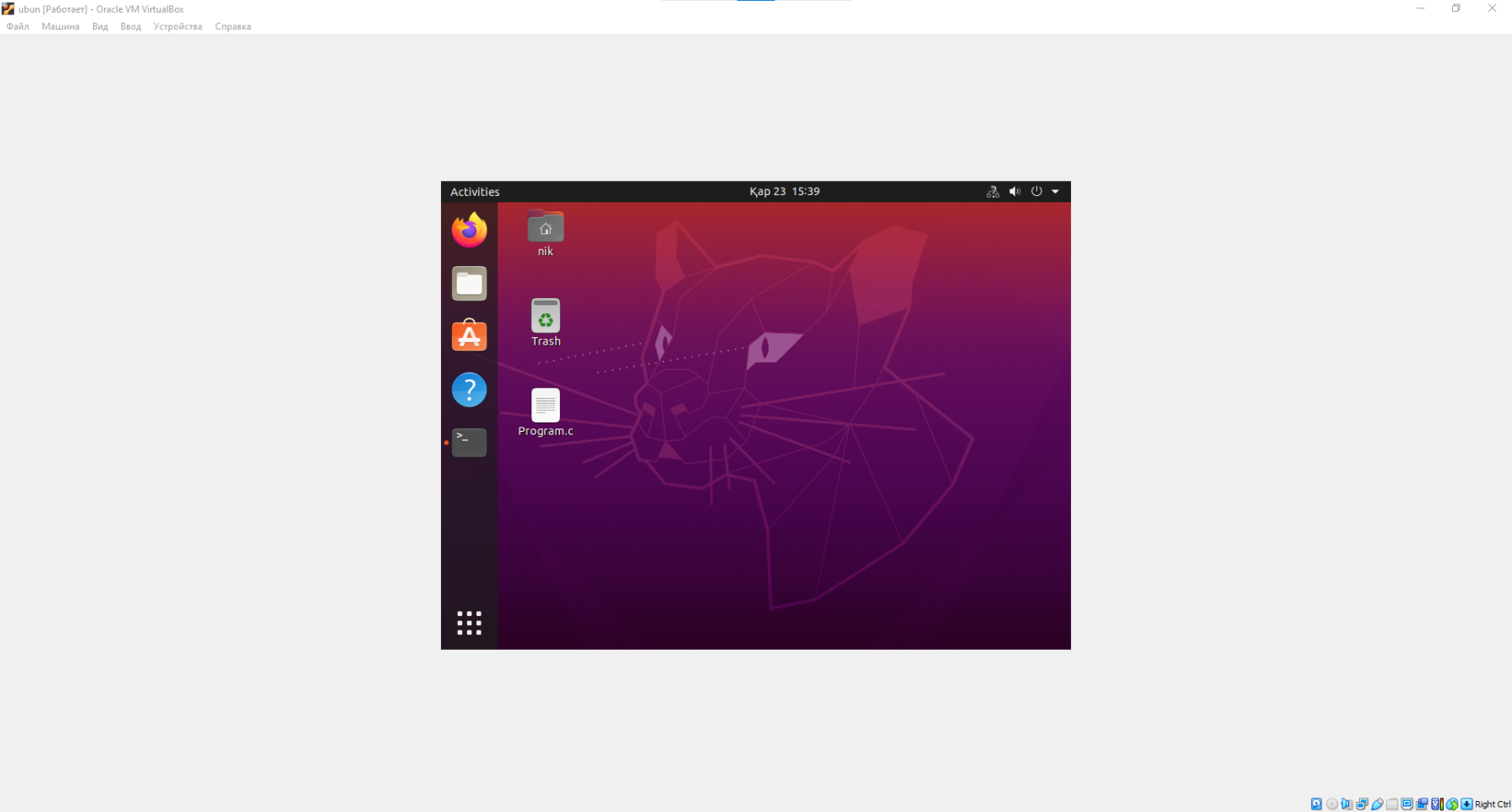


Рис.43 Файл на рабочем столе

Исходный код

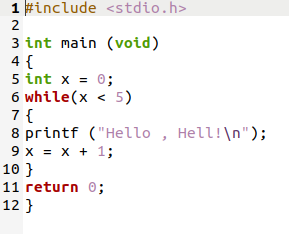


Рис.44 Исходный код

Код в файле для компиляции

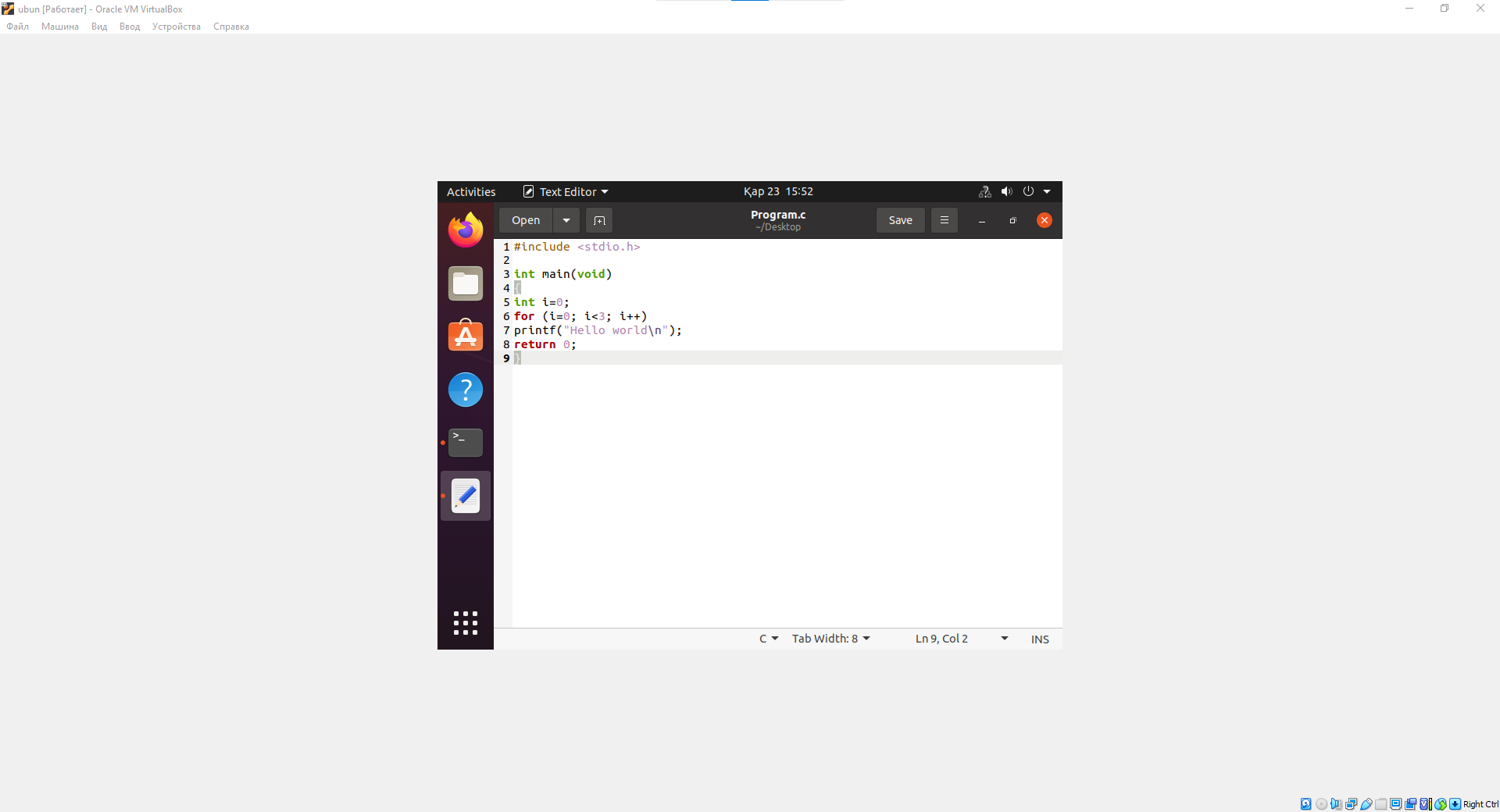


Рис.45 Код

Компиляция файла с кодом



Рис.46 Компиляция

Запуск скомпилированного файла



Рис.47 Скомпилированный файл

# ЗАКЛЮЧЕНИЕ

Целью учебной практики по ПМ.03 «Ревьюирирование программных продуктов» являлось, научиться выполнять прямое и обратное проектирование ПО, овладеть навыками приостановки и возобновления работы с рабочими задачами иинспекцией кода в Visual Studio, совместно работать над проектом в системе контроля версий GIT.

В ходе выполнения заданий на учебную практику было выполнено:

Изучили термины (репозиторий, дизассемблер, GitHub, radare2, компилятор, GCC, ПО, обратная разработка, Linux)

Измерили производительность приложения посредством анализа использования центрального процессора, по таким параметрам как: Ядро, IO, Файловая система. Проверили нагрузку центрально процессора с помощью разных фильтров отображения: вызывающий/вызываемый, дерево вызовов, функции.

установили программное обеспечение (Linux, компилятор GCC, radare2, iaito) для обратного проектирования,

Также мы выполнили обратное проектирования. Работали с файлами и компилировали через Терминал.

Мы выполнили дизассемблирование и работали с графическим интерфейсом Iaito, Radare2, и через них изменяли неправильный код.

# ЛИТЕРАТУРА

1. 3.Золотов, С. Ю. Проектирование информационных систем [Электронный ресурс] / С.Ю. Золотов. - Томск: Эль Контент, 2013. - 88 с.
2. Карпенков, С. Х. Технические средства информационных технологий [Электронный ресурс] / С.Х. Карпенков. - 3-е изд., испр. и доп. - М.|Берлин: Директ-Медиа, 2015. - 376 с.
3. Коноплева, И. А. Информационные технологии [Электронный ресурс] / И.А. Коноплева. - 2-е изд., перераб. и доп. - Москва: Проспект, 2014. - 328 с.
4. Корячко, В. П. Процессы и задачи управления проектами информационных систем [Электронный ресурс] / В.П. Корячко. - Москва: Горячая линия - Телеком, 2014. - 376 с.
5. Ланских, Юрий Владимирович Предметно-ориентированные информационные системы [Электронный ресурс] : учеб. пособие для студентов направления 09.03.02, 10.03.01,
6. Проектирование информационных систем. Лекция 1. Презентация [Электронный ресурс]. - Москва: Национальный Открытый Университет «ИНТУИТ», 2014. - 27 с.
7. Советов, Борис Яковлевич. Информационные технологии [Электронный ресурс]: учебник / Б. Я. Советов, В. В. Цехановский. - 6-е изд. - Москва: Юрайт, 2015. - х эл. опт. диск (CD-ROM)
8. Советов, Борис Яковлевич. Информационные технологии [Электронный ресурс]: учебник / Б. Я. Советов, В. В. Цехановский. - 6-е изд. - Москва: Юрайт, 2015. - х эл. опт. диск (CD-ROM)
9. Страбыкин, Дмитрий Алексеевич. Организация ЭВМ: лабораторный практикум на компьютерах: учеб. пособие для студентов направления подготовки 09.03.01 (230100.62) / Д. А. Страбыкин; ВятГУ, ФАВТ, каф. ЭВМ. - 3-е изд., перераб. и доп. - Киров: [б. и.], 2013. - 62 с.

# Приложение 1

**Ссылки на репозиторий с исходными файлами**

https://github.com/dick-hub/PRAKTICUS - Изменение производительности приложения посредством анализа использования ЦП