Отчёт по лабораторной работе №2

Дисциплина: архитектура компьютера

Челухаев Кирилл Александрович

Содержание

# 1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

# 2 Задание

Настройка GitHub. Базовая настройка Git. Создание рабочего пространства и репозитория курса на основе шаблона. Создание репозитория на основе шаблона Настройка каталога курса. Задания для самостоятельной работы

# 3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений). Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту и при необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий.

# 4 Выполнение лабораторной работы

## 4.1 Настройка GitHub

Я создал учетную запись на GitHub (рис. 1).

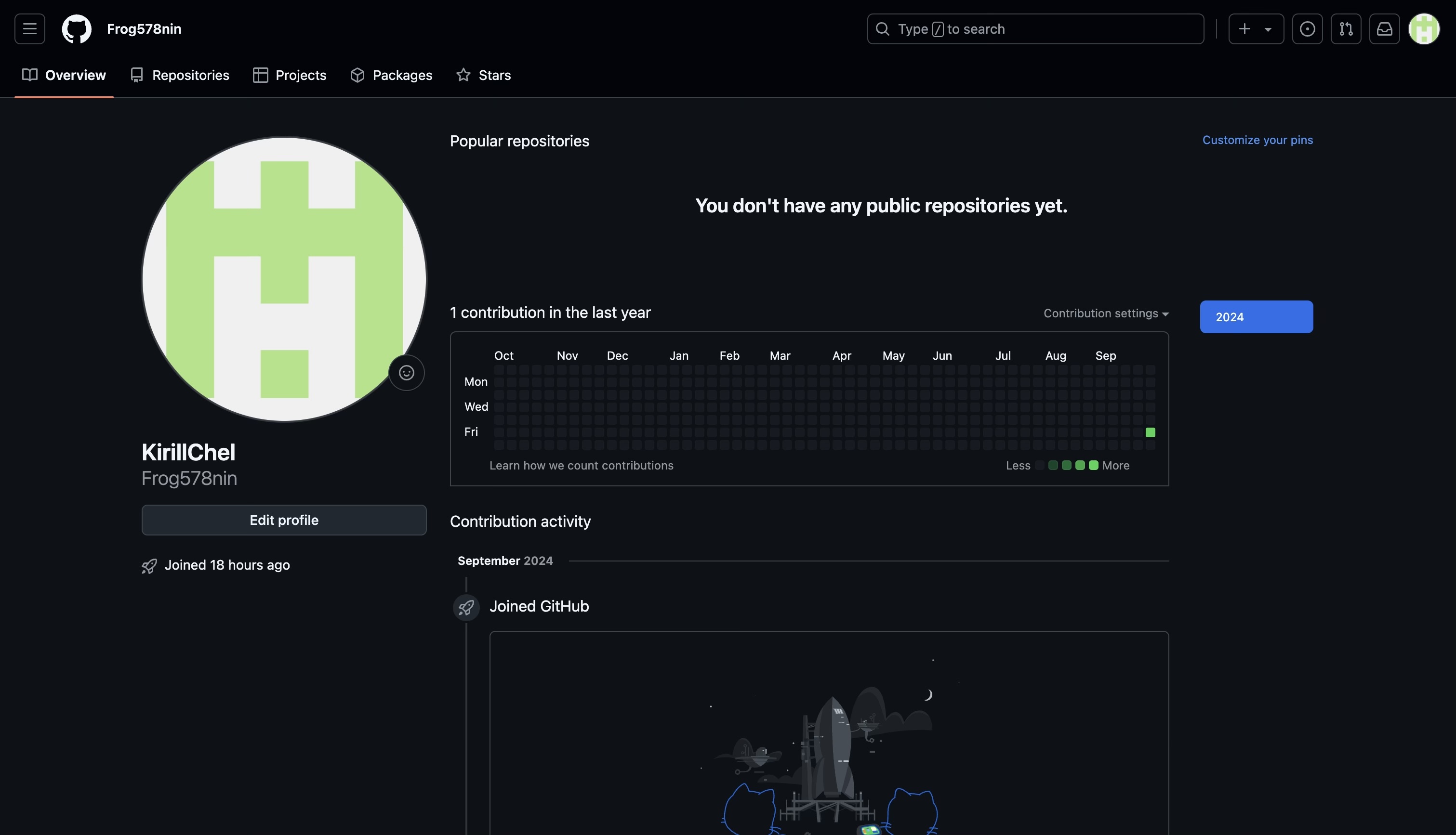


Рис. 1: Учетная запись GitHub

## 4.2 Базовая настройка Git

Я открыл терминал, где нахожусь в домашней директории (рис. 2).

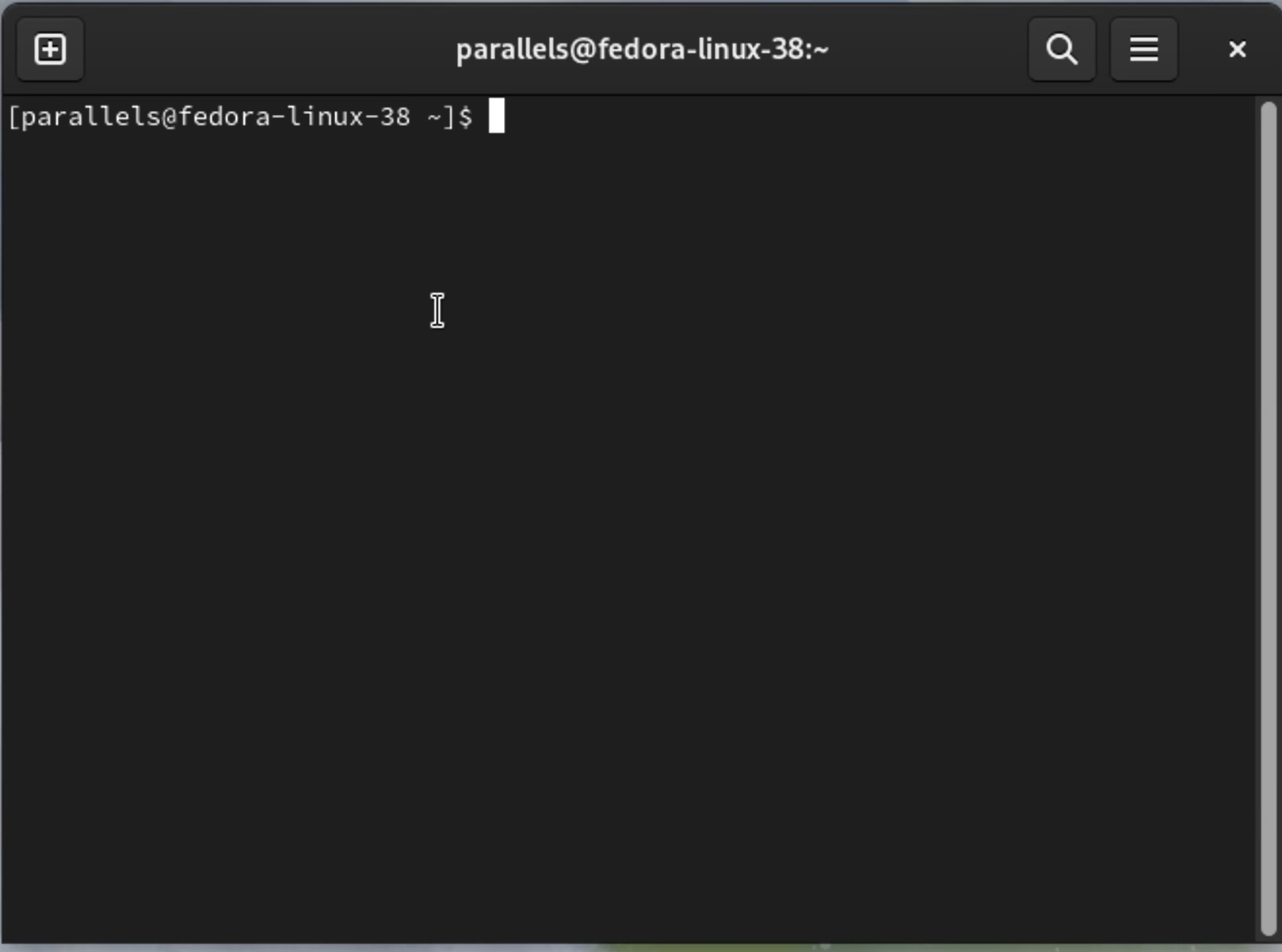


Рис. 2: Домашняя директория

Далее я ввел команды git config –global user.name, указав свое имя, и команду git config –global user.email, указав свою корпоративную почту. (рис. 3).

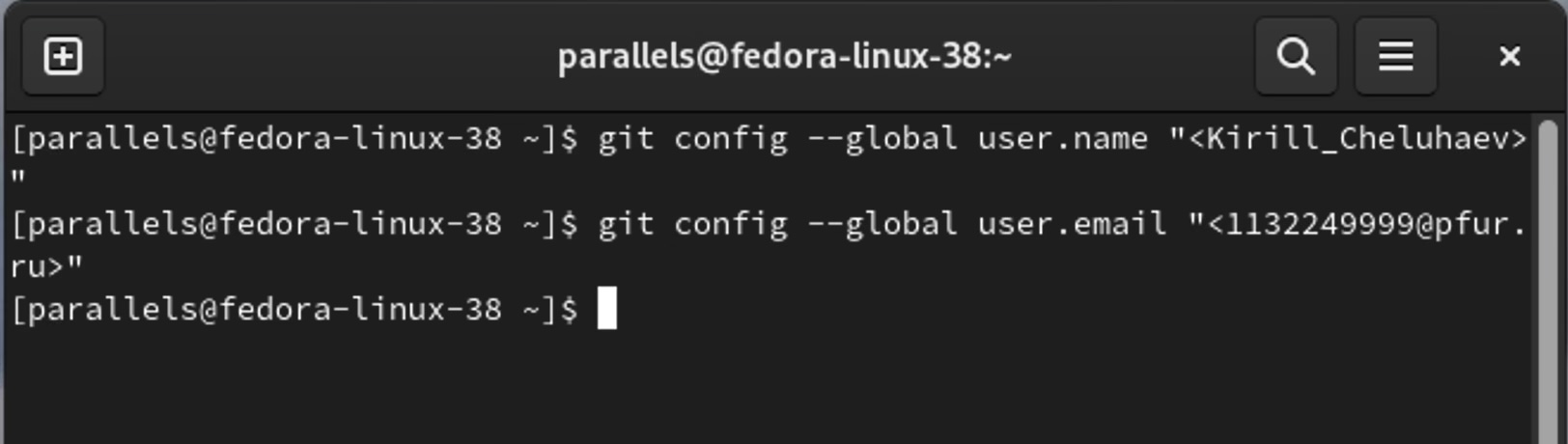


Рис. 3: Предварительная конфигурация

Теперь я настроил utf-8 в выводе сообщений в git (рис. 4).

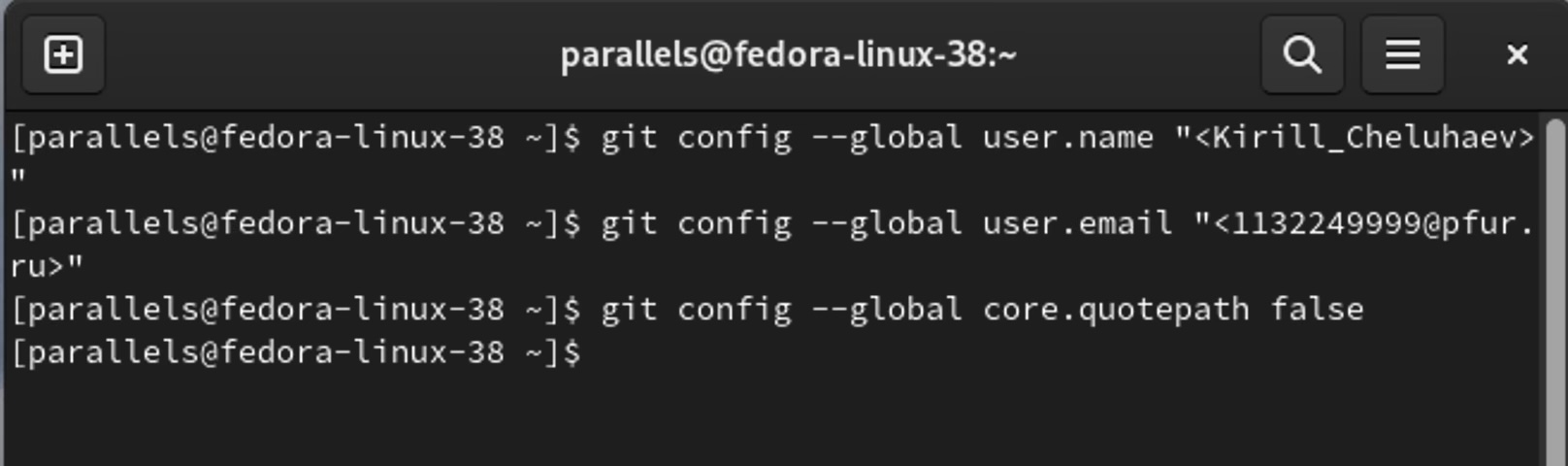


Рис. 4: Настройка utf-8

Далее я задал имя главной ветки - master. (рис. 5).

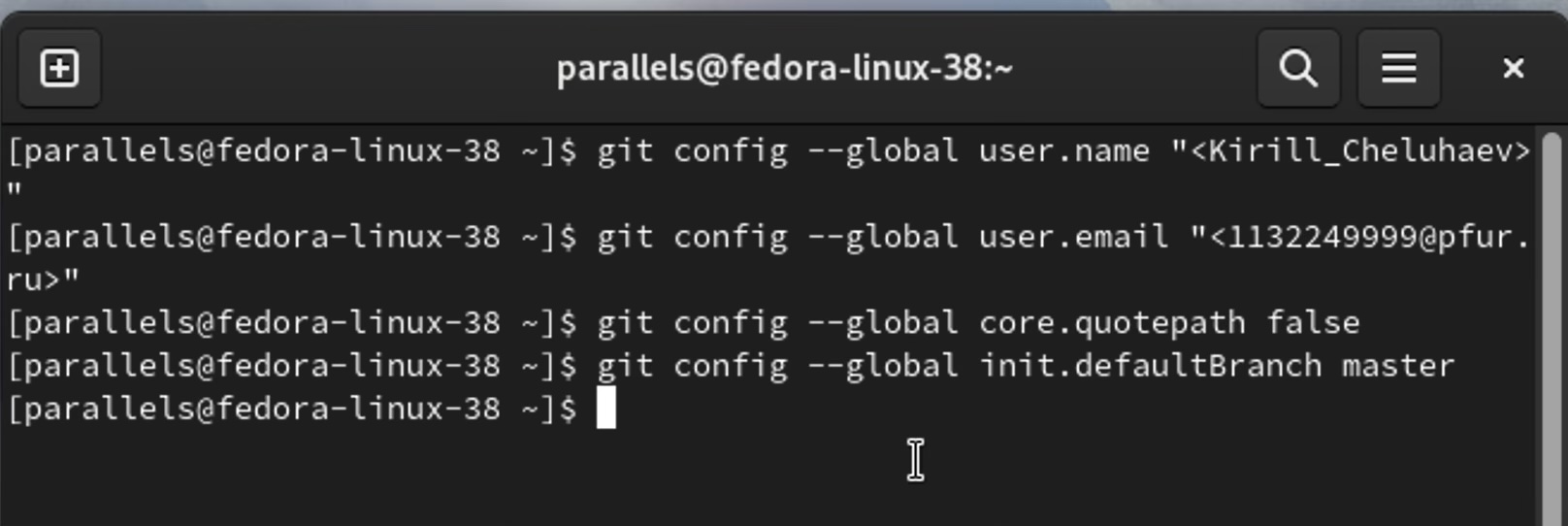


Рис. 5: Главная ветвь master

Теперь я задаю параметр autocrlf (рис. 6).

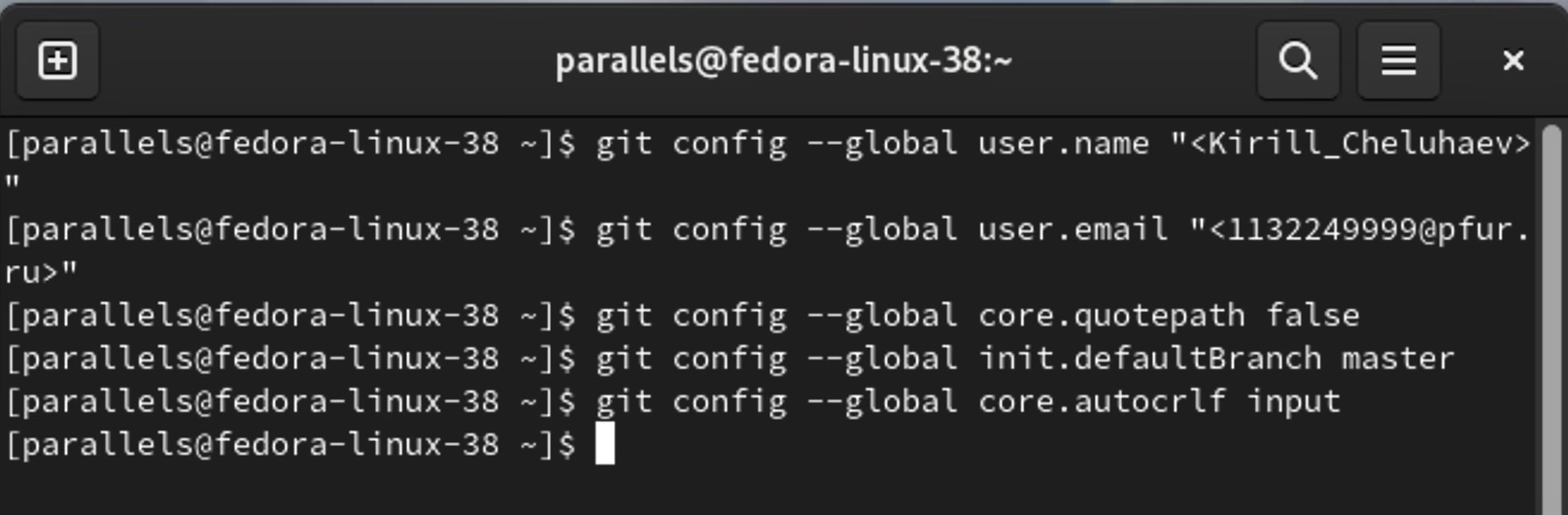


Рис. 6: Параметр autocrlf

Далее я задал параметр safecrlf (рис. 7).

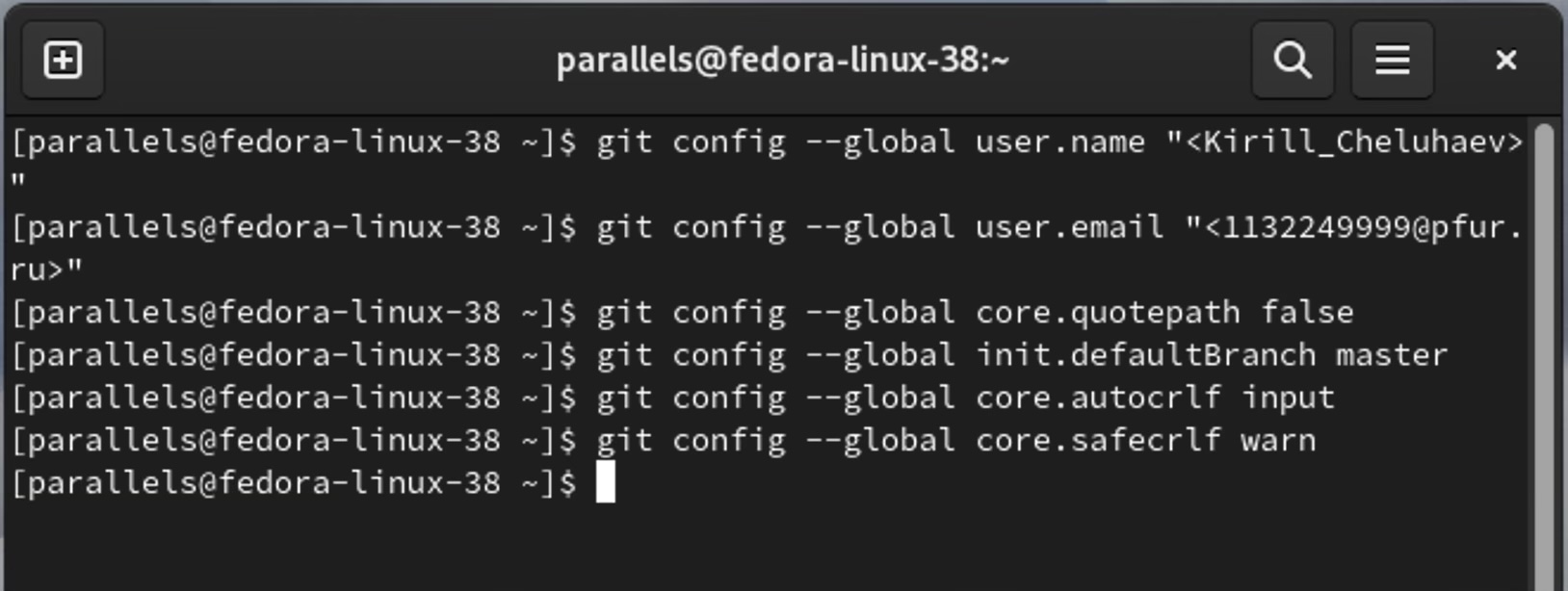


Рис. 7: Параметр safecrlf

## 4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев я создал пару ключей (приватный и открытый)(рис. 8).

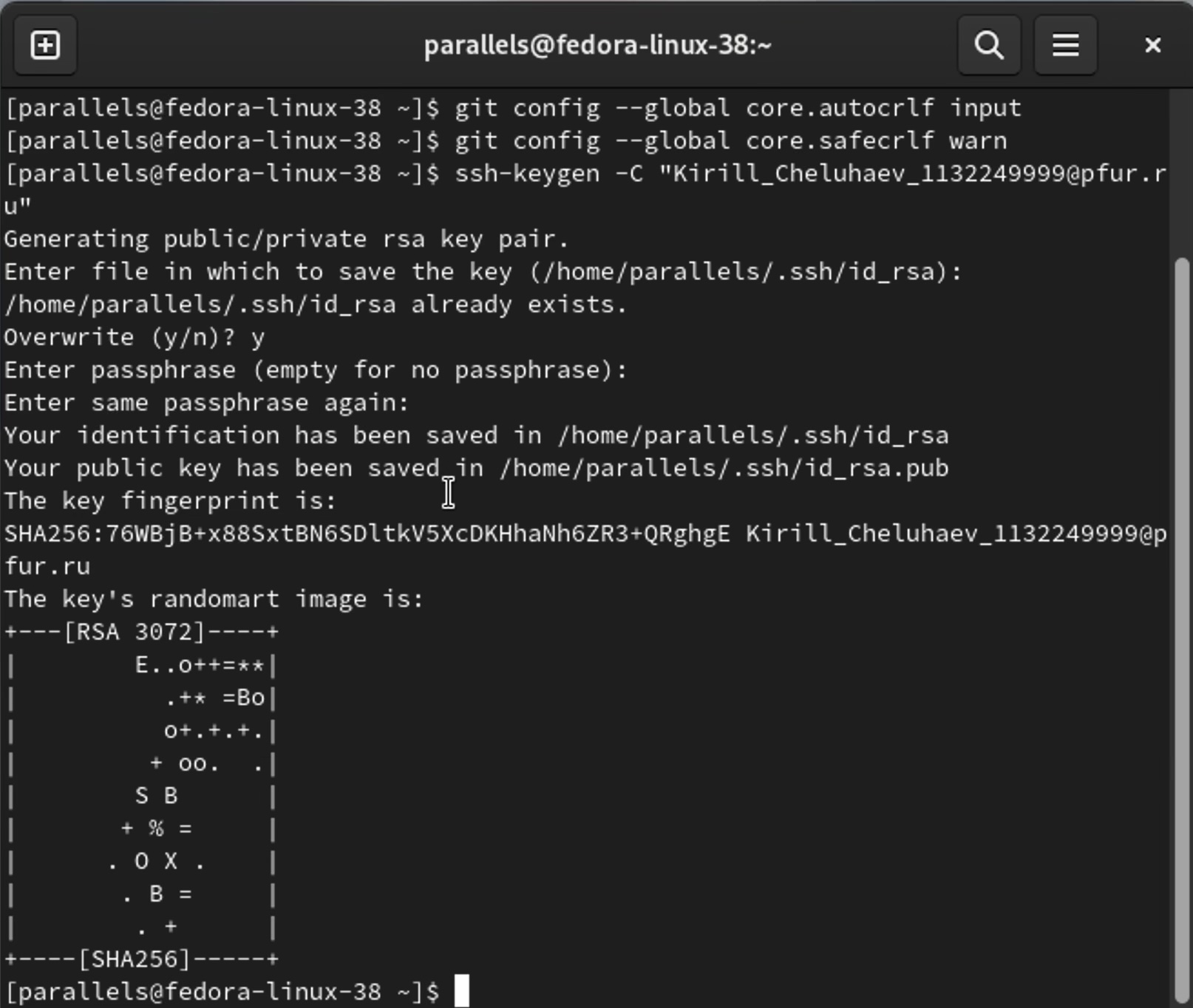


Рис. 8: Создание SSH-ключа

Далее я скопировал сгенерированный ключ в буфер обмена и загрузил его на сайт GitHub.com под именем Title (рис. 9) (рис. 10).

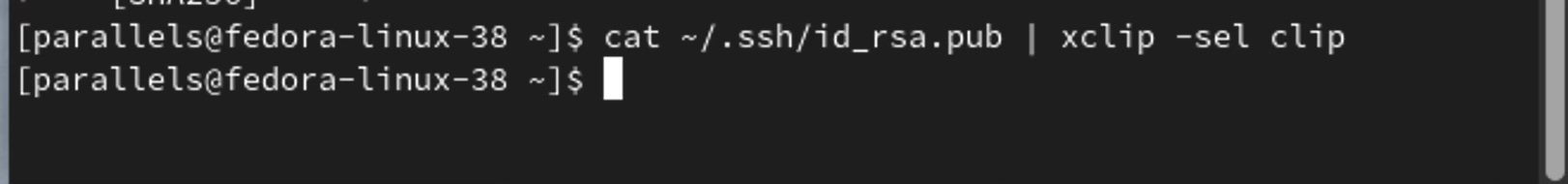


Рис. 9: Копирование SSH-ключа

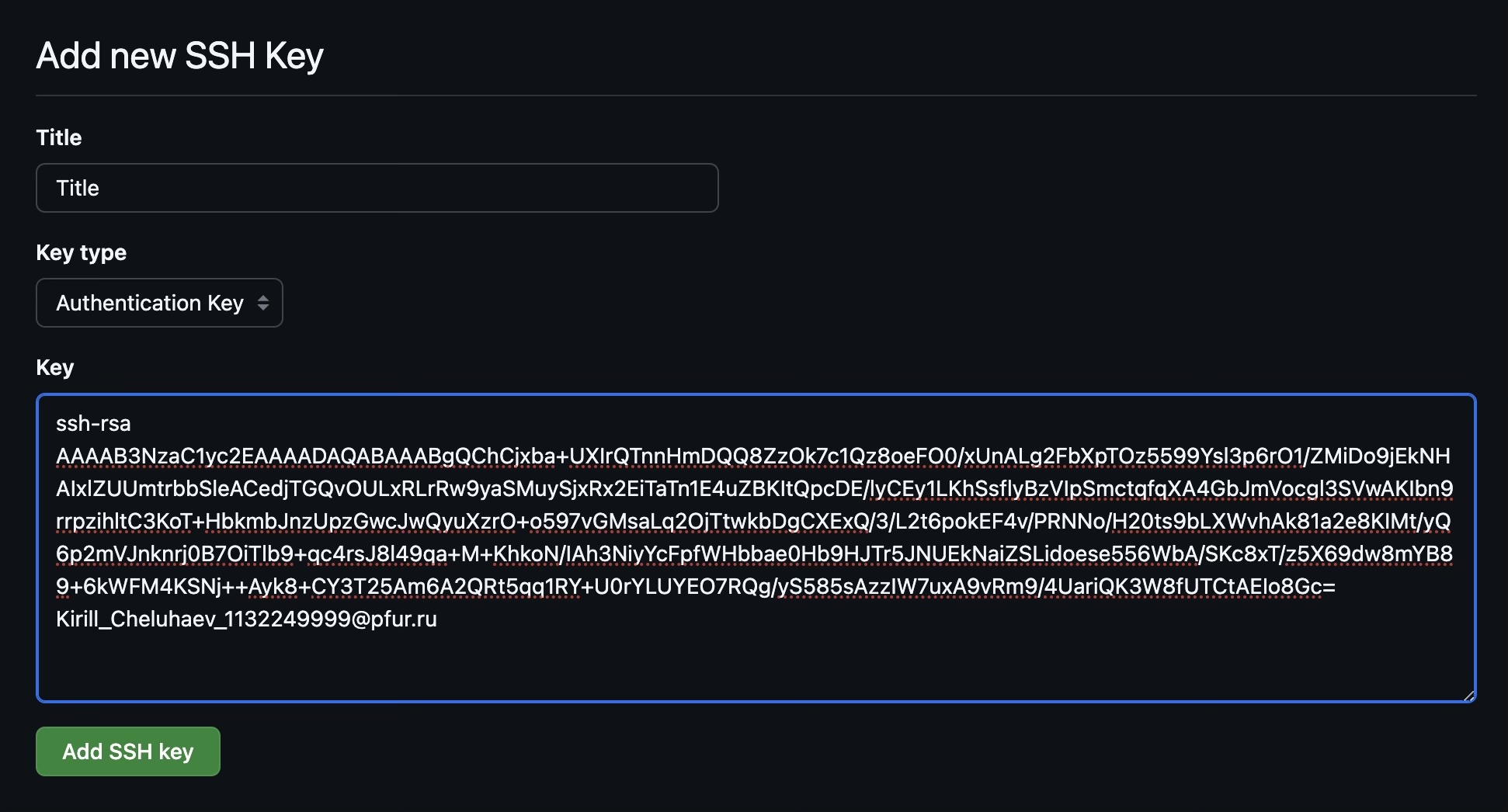


Рис. 10: Загрузка SSH-ключа на GitHub

## 4.4 Создание рабочего пространства и репозитория на основе шаблона

Я создал каталог для предмета “Архитектура компьютера” по шаблону (рис. 11)

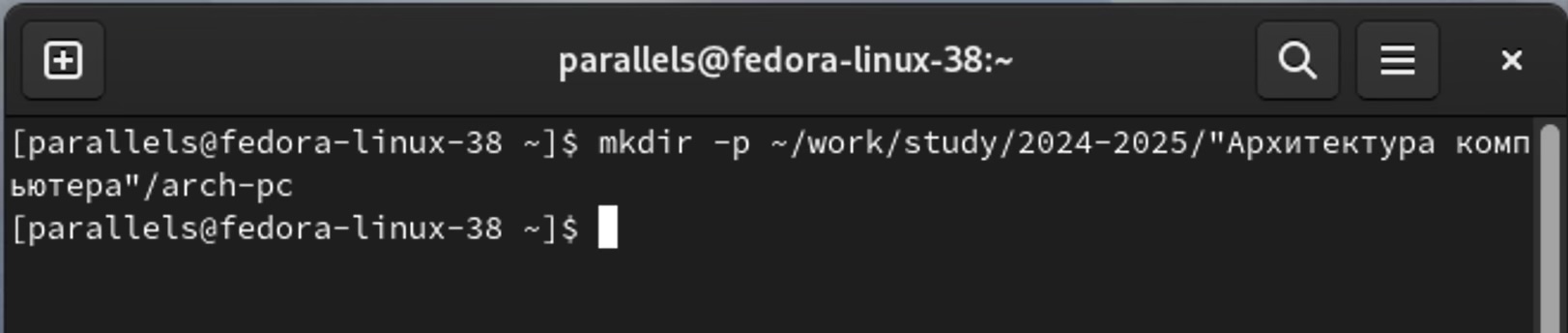


Рис. 11: Каталог “Архитектура компьютера”

## 4.5 Создание репозитория курса на основе шаблона

Я перешел на страницу репозитория с шаблоном курса и создал репозиторий с именем study\_2024-2025\_arch-pc (рис. 12) (рис. 13)

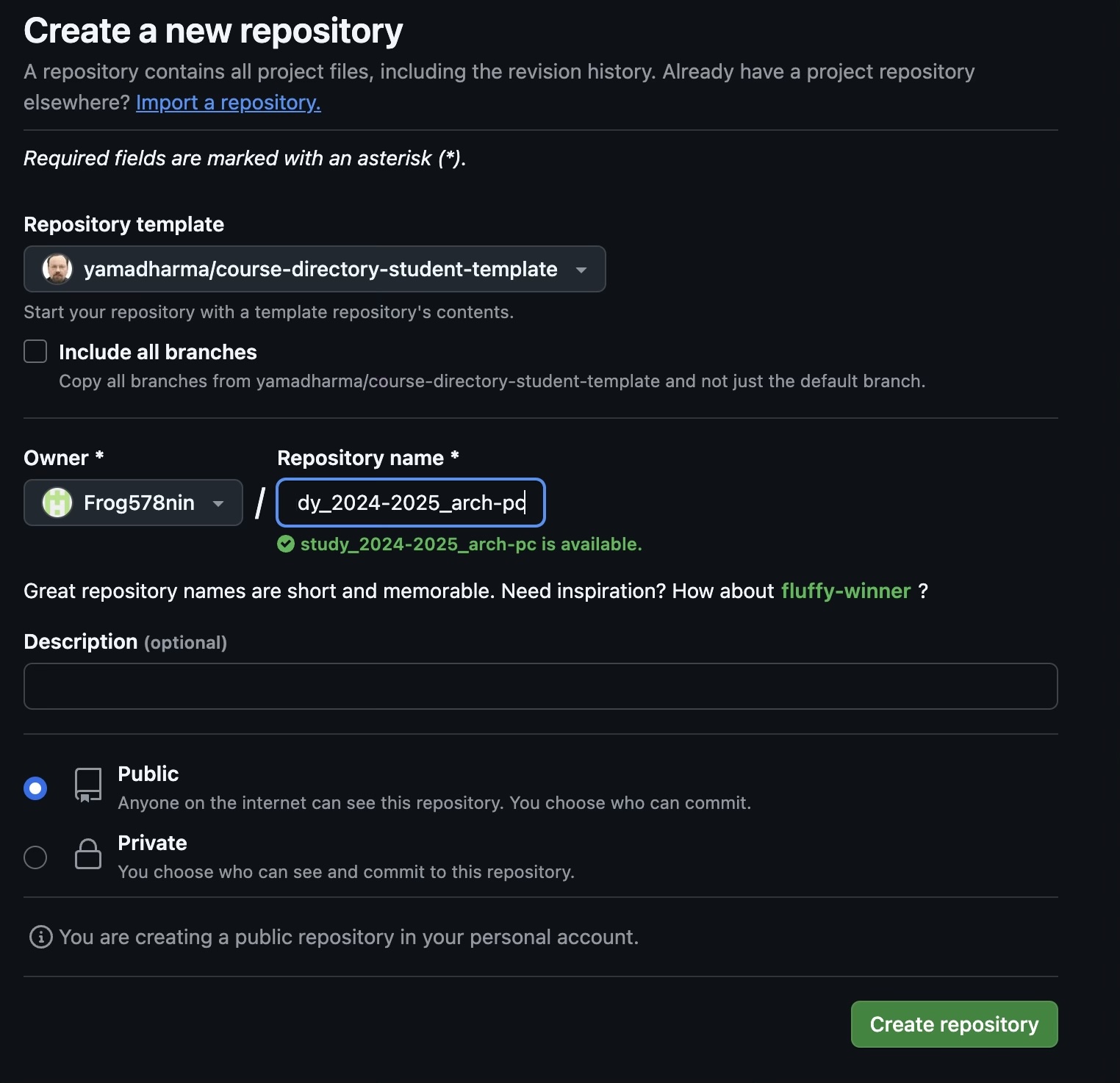


Рис. 12: Создание репозитория

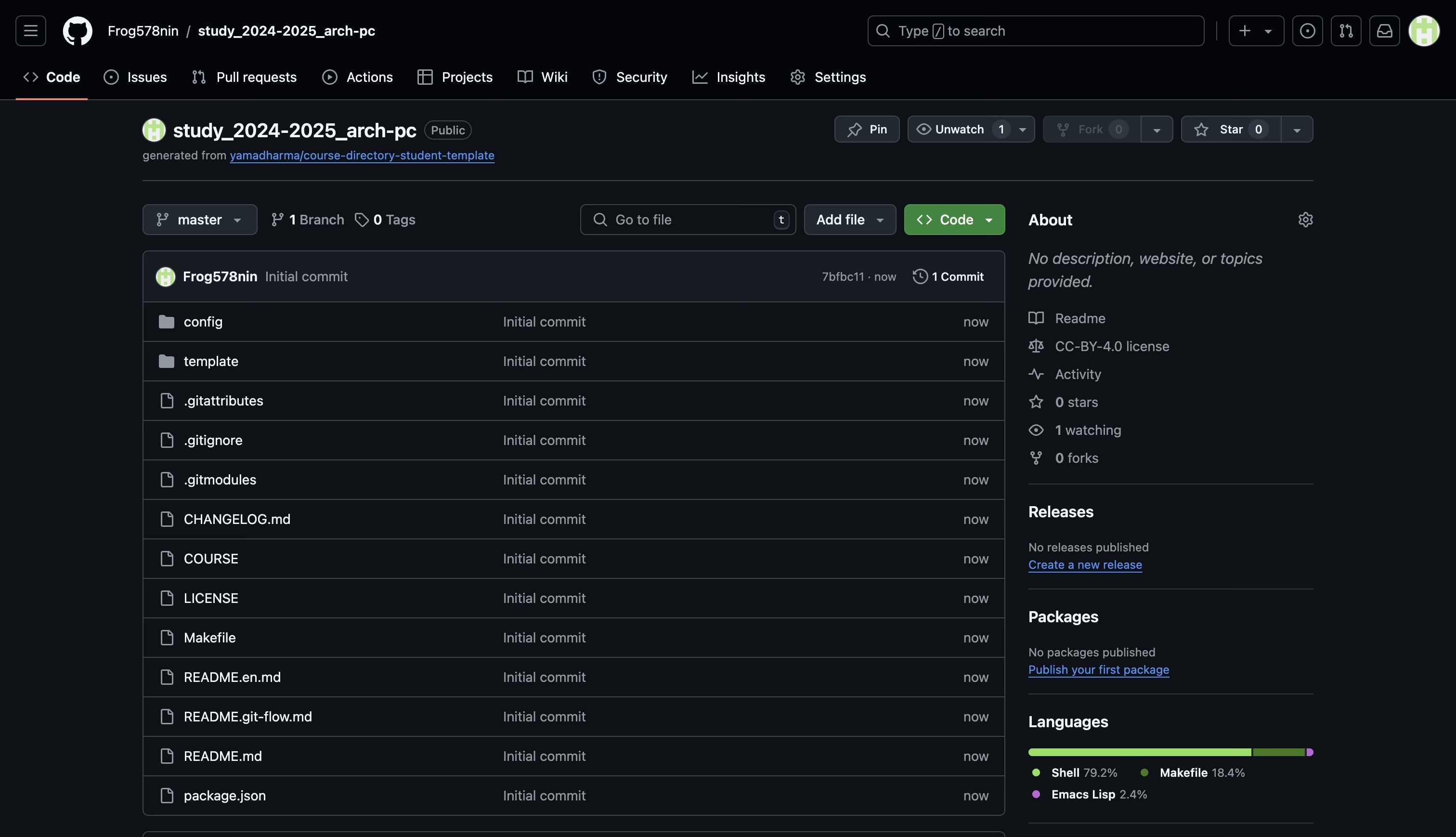


Рис. 13: Созданный репозиторий

Далее я открыл терминал и перешел в каталог курса (рис. 14)

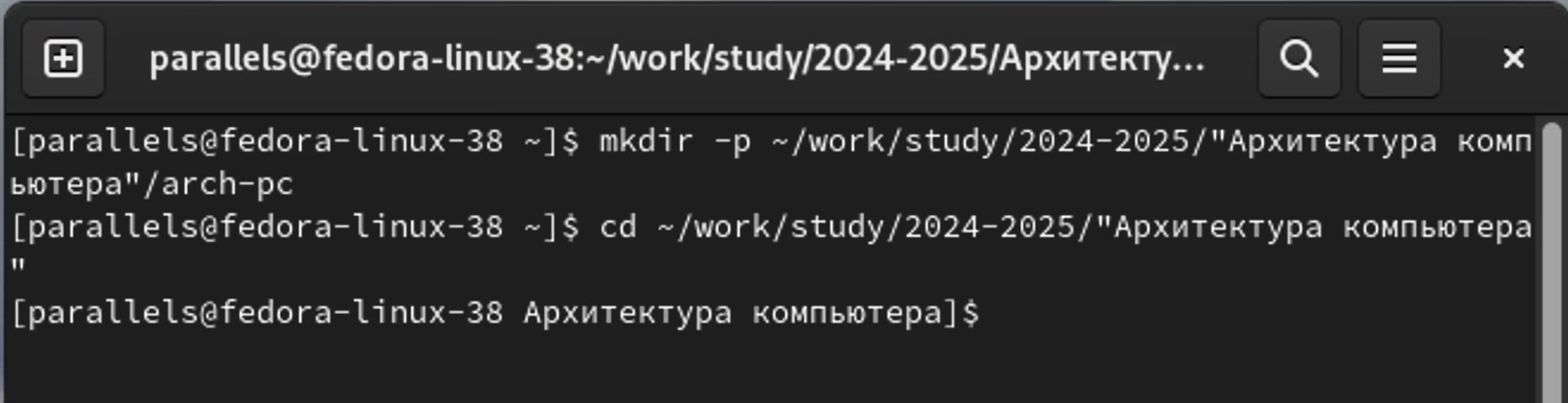


Рис. 14: Перемещение в новый каталог

Теперь я клонировал созданный репозиторий (рис. 15)

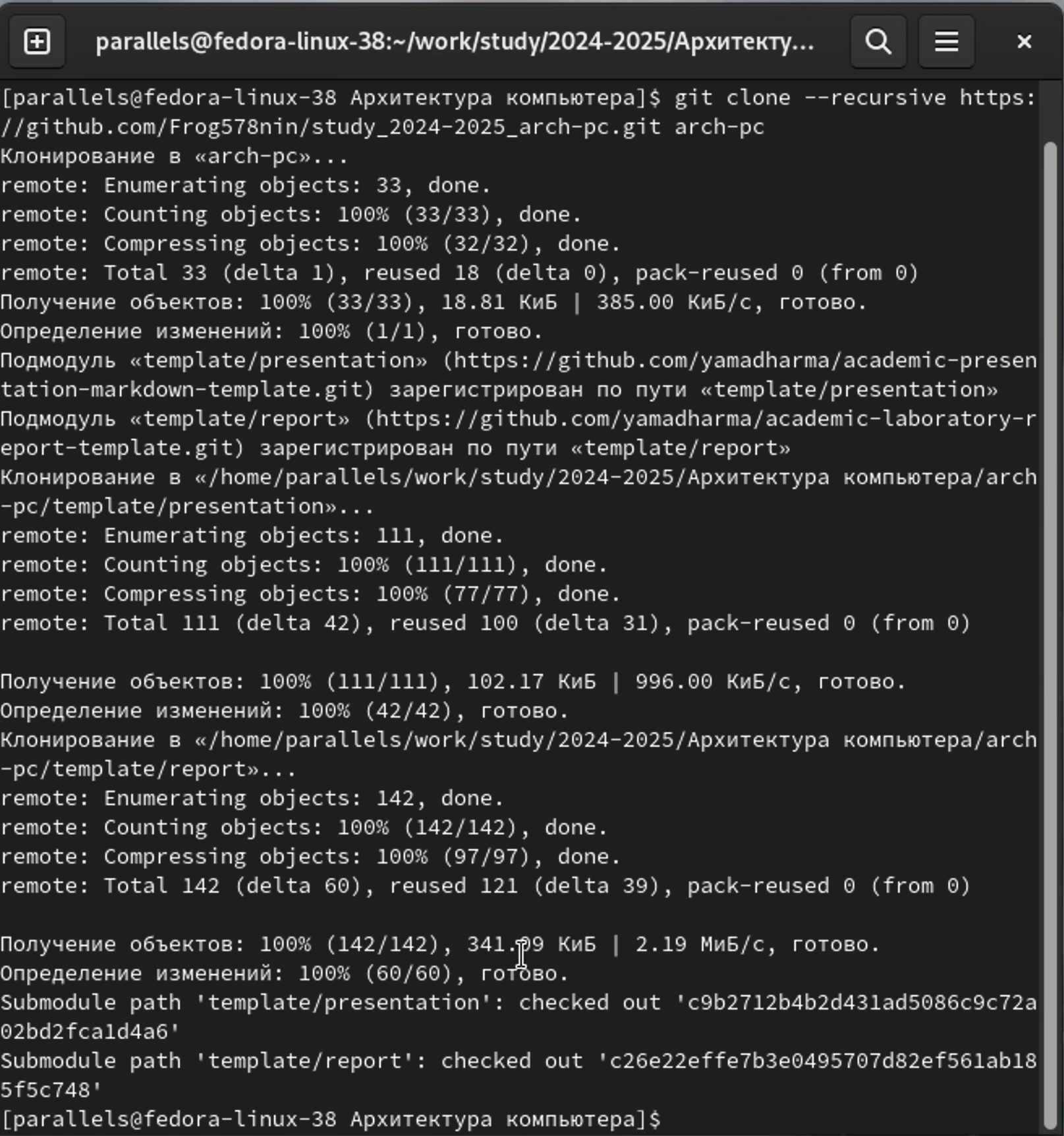


Рис. 15: Клонирование репозитория

## 4.6 Настройка каталога курса

Далее я перешел в каталог курса и удалил все лишние файлы командой rm (рис. 16)(рис. 17)

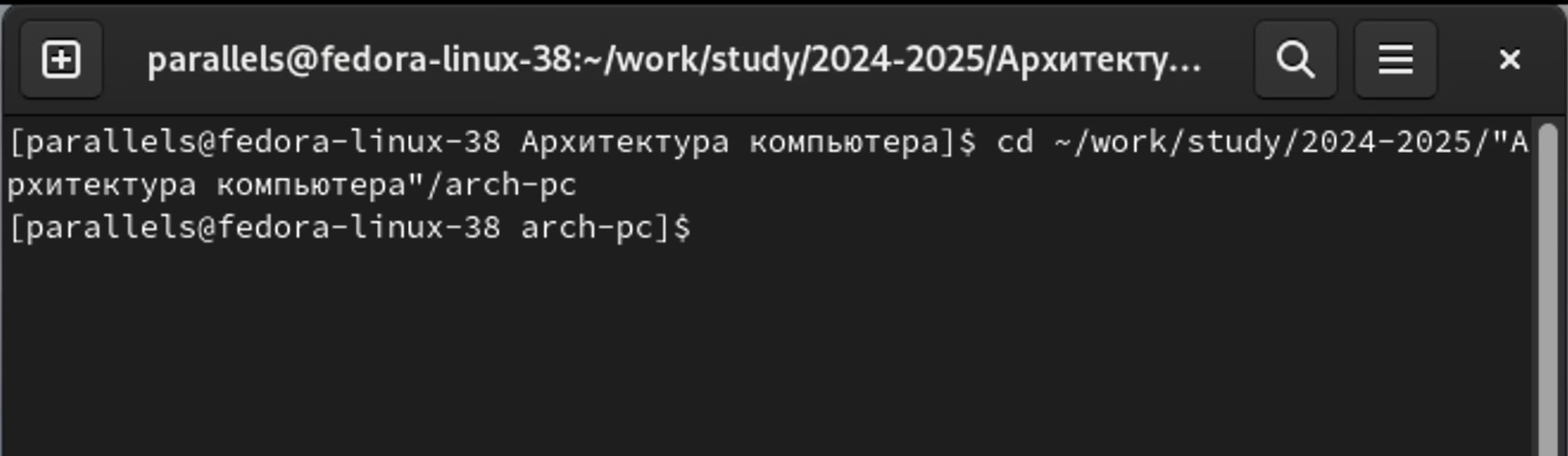


Рис. 16: Переход в каталог курса

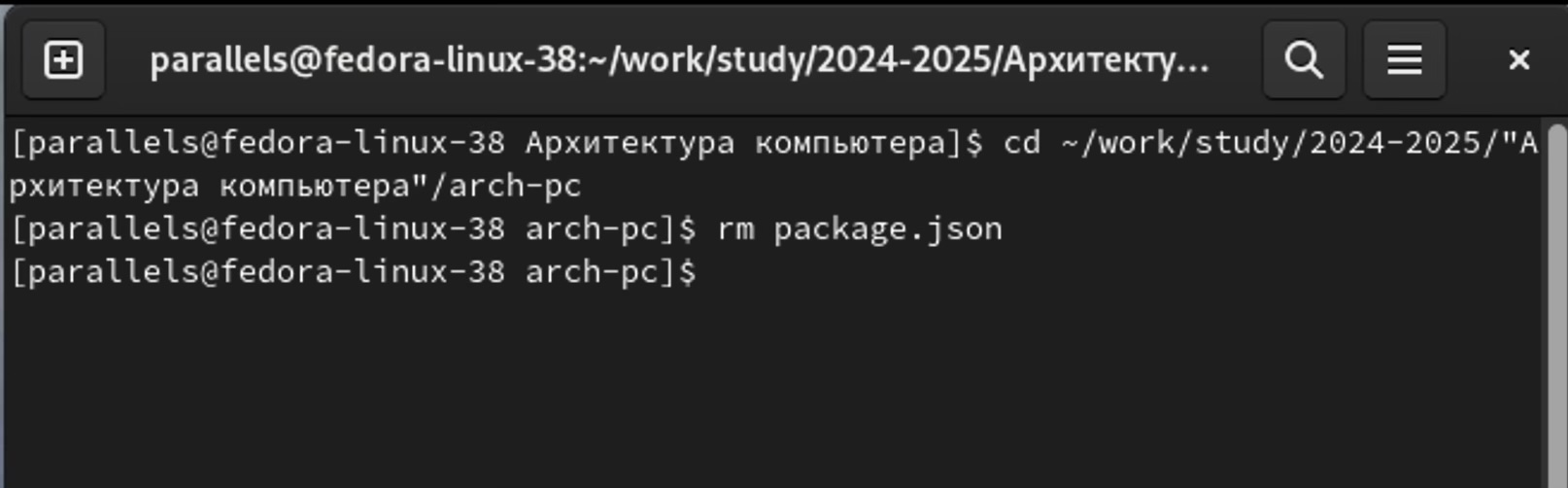


Рис. 17: Удаление лишних файлов

Теперь я создал все необходимые каталоги и отправил файлы на сервер (рис. 18) (рис. 19)

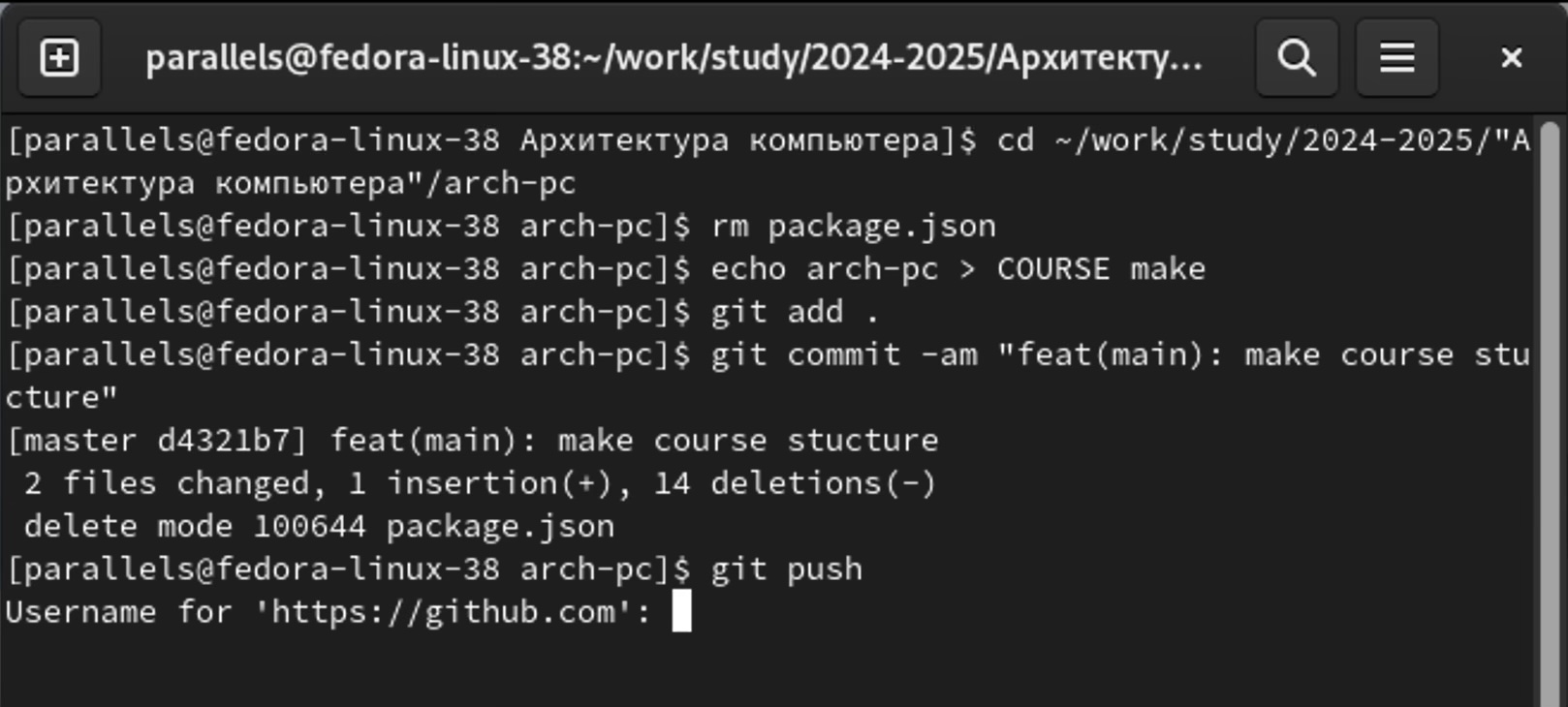


Рис. 18: Создание каталогов

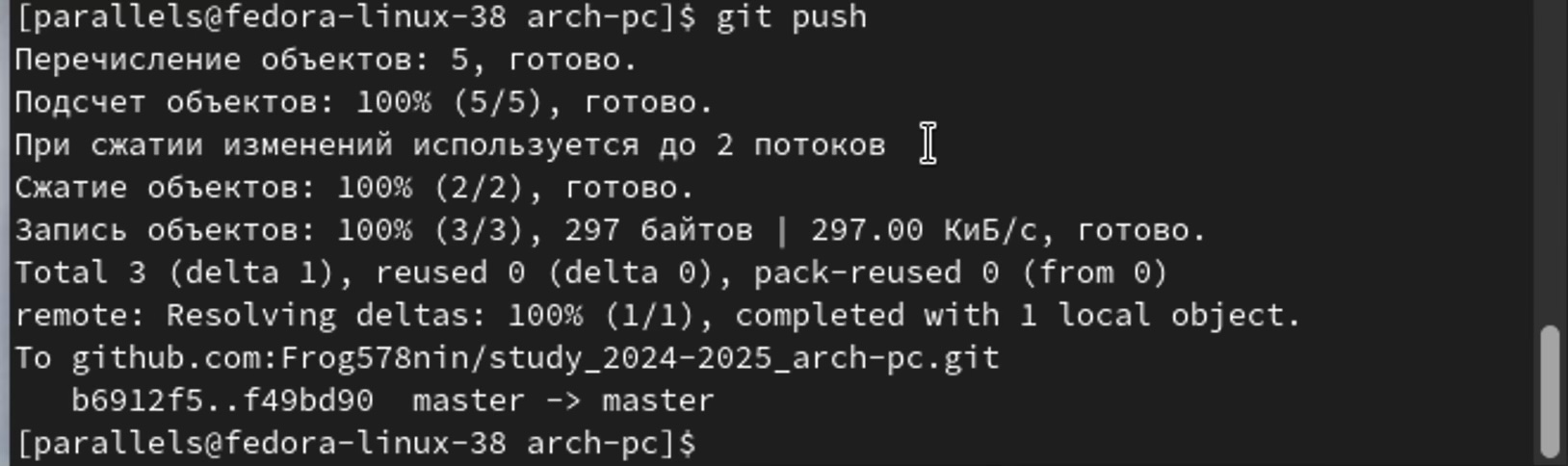


Рис. 19: Отправление файлов на сервер

## 4.7 Выполнение заданий для самостоятельной работы

### 4.7.1 Создать отчет по выполненной лабораторной работы в соответствующем каталоге рабочего пространства

Я создал нужный мне каталог labs/lab02/report в каталоге рабочего пространства (рис. 20)

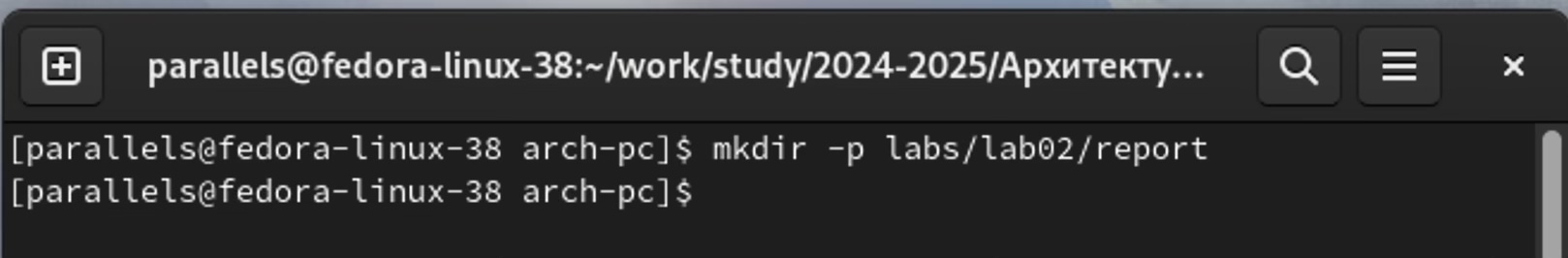


Рис. 20: Создание нужного каталога

Далее я переместил лабораторную работу 02 в этот каталог при помощи команды mv и проверил корректность выполнения команд при помощи команды ls (рис. 21) (рис. 22)

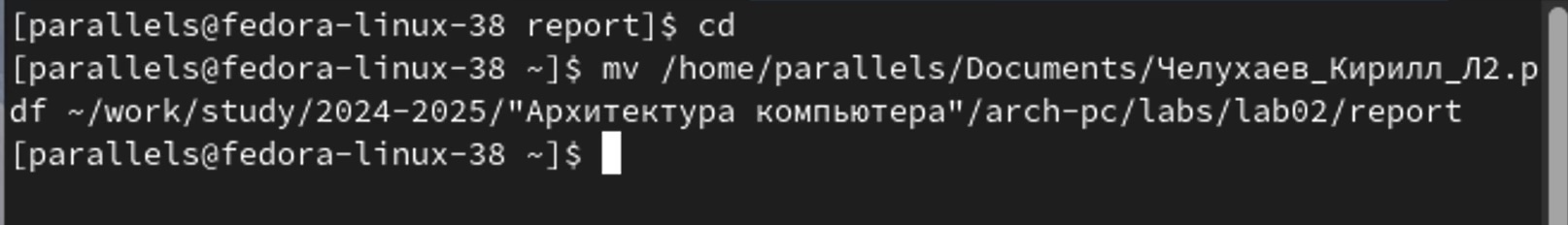


Рис. 21: Перемещение отчета

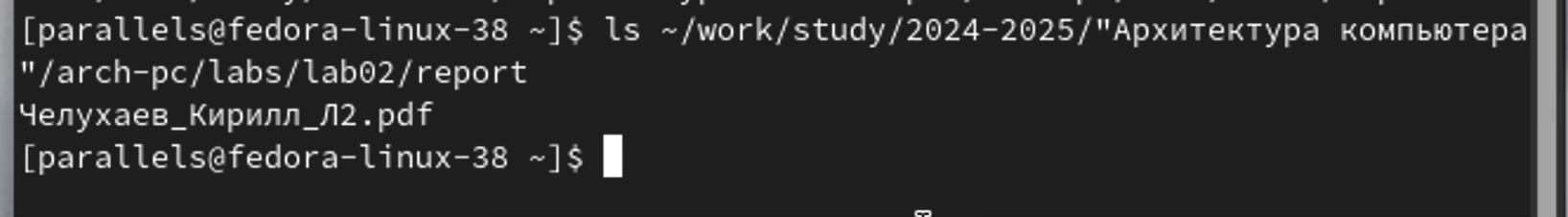


Рис. 22: Корректное выполнение команды mv

### 4.7.2 Перенести предыдущие отчеты по лабораторным работам в соответствующие каталоги рабочего пространства

Я создал нужный мне каталог labs/lab01/report в каталоге рабочего пространства (рис. 23)

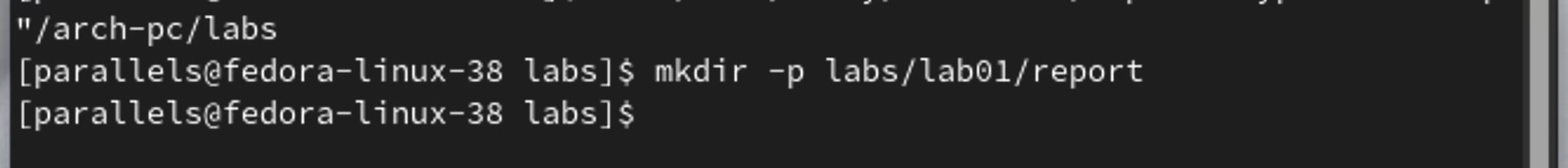


Рис. 23: Создание каталога для прошлого отчета

Далее я перенес файл по лабораторной работе 01 в нужный каталог рабочего пространства и проверил корректность выполнения команды при помощи команды ls (рис. 24)

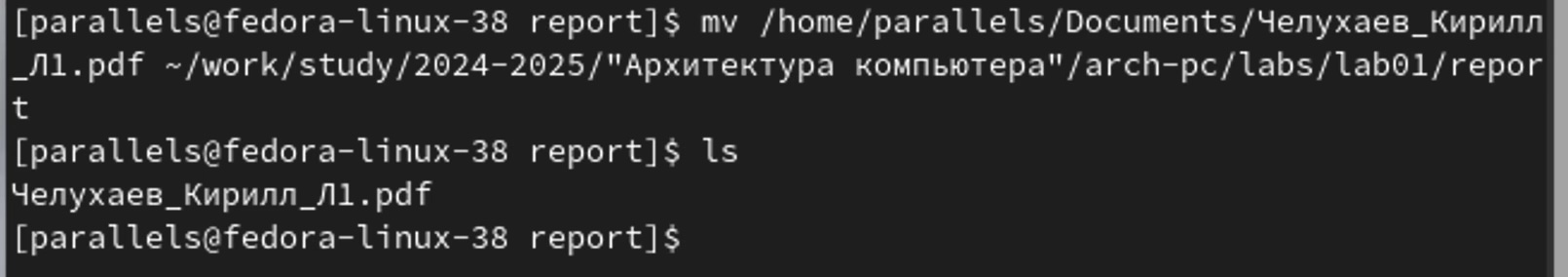


Рис. 24: Перенос лабораторной работы 01

### 4.7.3 Загрузка файлов на GitHub

Я загрузил все файлы на GitHub (рис. 25)

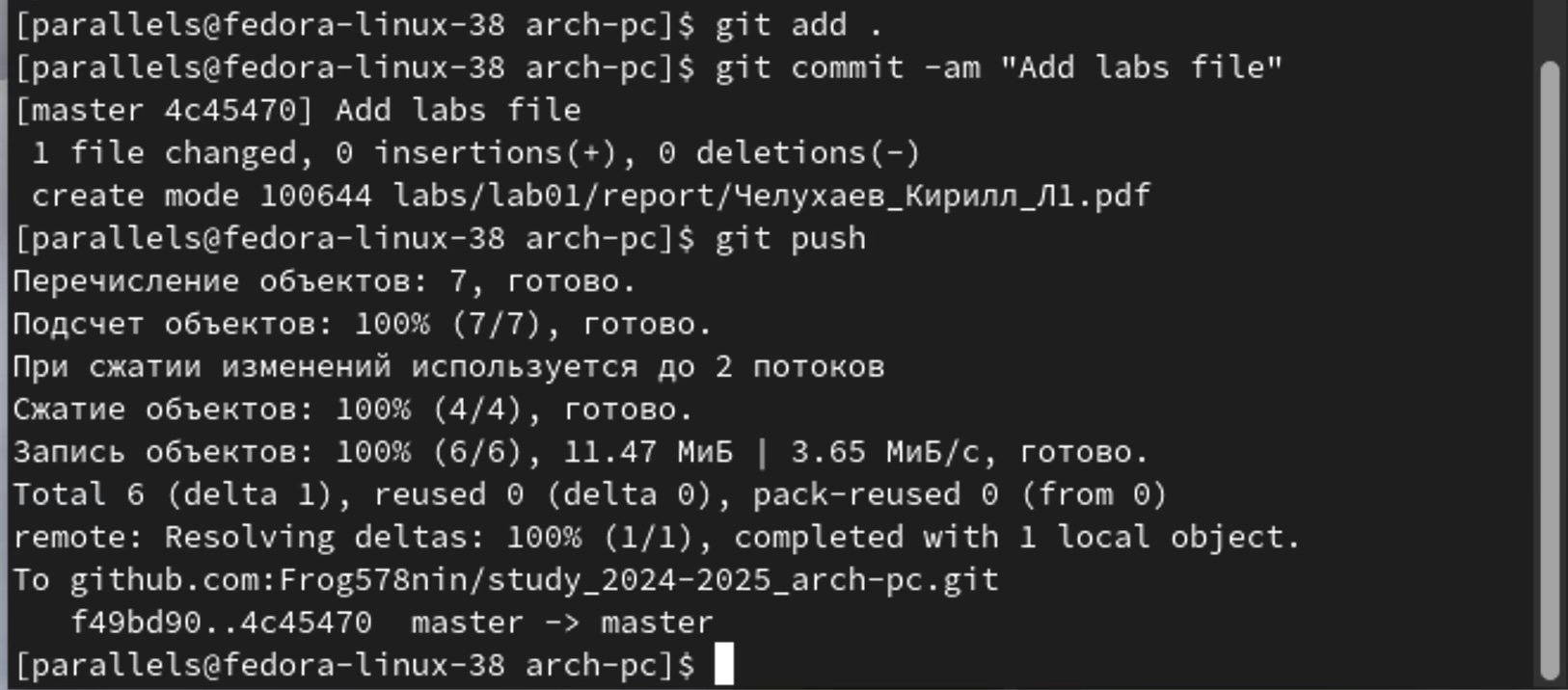


Рис. 25: Загрузка файлов на GitHub

# 5 Выводы

Таким образом я изучил идеологию и применение средств контроля версий и получил практические навыки по работе с системой git.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. —(Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science). ::: {#refs} :::