Отчет по лабораторной работе №2

Архитектура компьютера и операционные системы

Александр Дмитриевич Собко

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 git, создание ssh-ключа, его подключение и практика основных команд

# 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.

# 4 Выполнение лабораторной работы

1. Настройка github Существует несколько доступных серверов репозиториев с возможностью бесплатного размещения данных. Например, http://bitbucket.org/, https://github.com/ и https://gitflic.ru. Для выполнения лабораторных работ предлагается использовать Github. Создайте учётную запись на сайте https://github.com/ и заполните основные данные.
2. Базовая настройка git Сначала сделаем предварительную конфигурацию git. Откройте терминал и введите следующие команды, указав имя и email владельца репозитория: git config –global user.name “” git config –global user.email “[work@mail](mailto:work@mail)”

Рисунок1

Рис. 1: Рисунок1

Настроим utf-8 в выводе сообщений git: git config –global core.quotepath false

Рисунок2

Рис. 2: Рисунок2

Зададим имя начальной ветки (будем называть её master): git config –global init.defaultBranch master

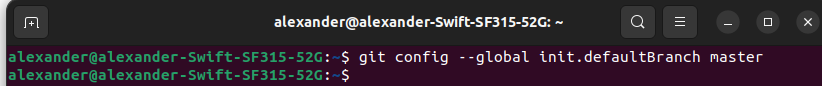


Рис. 3: Рисунок3

Параметр autocrlf: git config –global core.autocrlf input

Рисунок4

Рис. 4: Рисунок4

Параметр safecrlf: git config –global core.safecrlf warn

Рисунок5

Рис. 5: Рисунок5

1. Создание SSH ключа Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C “Имя Фамилия [work@mail](mailto:work@mail)”

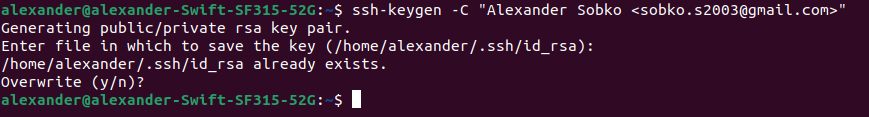


Рис. 6: Рисунок6

Ключи сохранятся в каталоге ~/.ssh/. Далее необходимо загрузить сгенерированный открытый ключ. Для этого зайти на сайт http://github.org/ под своей учётной записью и перейти в меню Setting . После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key . Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id\_rsa.pub | xclip -sel clip вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title)

Рисунок7

Рис. 7: Рисунок7

1. Создание рабочего пространства и репозитория курса на основе шаблона При выполнении лабораторных работ следует придерживаться структуры рабочего пространства. Рабочее пространство по предмету располагается в следующей иерархии: ~/work/study/ / / / Например, для 2023–2024 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) структура каталогов примет следующий вид: ~/work/study/ 2023–2024/ Архитектура компьютера/ arch-pc/ labs/ lab01/ lab02/ lab03/ … Каталог для лабораторных работ имеет вид labs. Каталоги для лабораторных работ имеют вид lab, например: lab01, lab02 и т.д. Название проекта на хостинге git имеет вид: study\_ Например, для 2023–2024 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) название проекта примет следующий вид: study 2023–2024 arch-pc Откройте терминал и создайте каталог для предмета «Архитектура компьютера»: mkdir -p /work/study/2023-2024/“Архитектура компьютера”
2. Создание репозитория курса на основе шаблона Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейдите на станицу репозитория с шаблоном курса https://github.com/yamadharma/course-directory-student-template. Далее выберите Use this template. В открывшемся окне задайте имя репозитория (Repository name) study\_2023-2024\_arch-pc и создайте репозиторий (кнопка Create repository from template). Откройте терминал и перейдите в каталог курса: cd /work/study/2023–2024/“Архитектура компьютера” клонируйте созданный репозиторий: git clone –recursive git@github.com:/study\_2023–2024\_arch-pc.git arch-pc

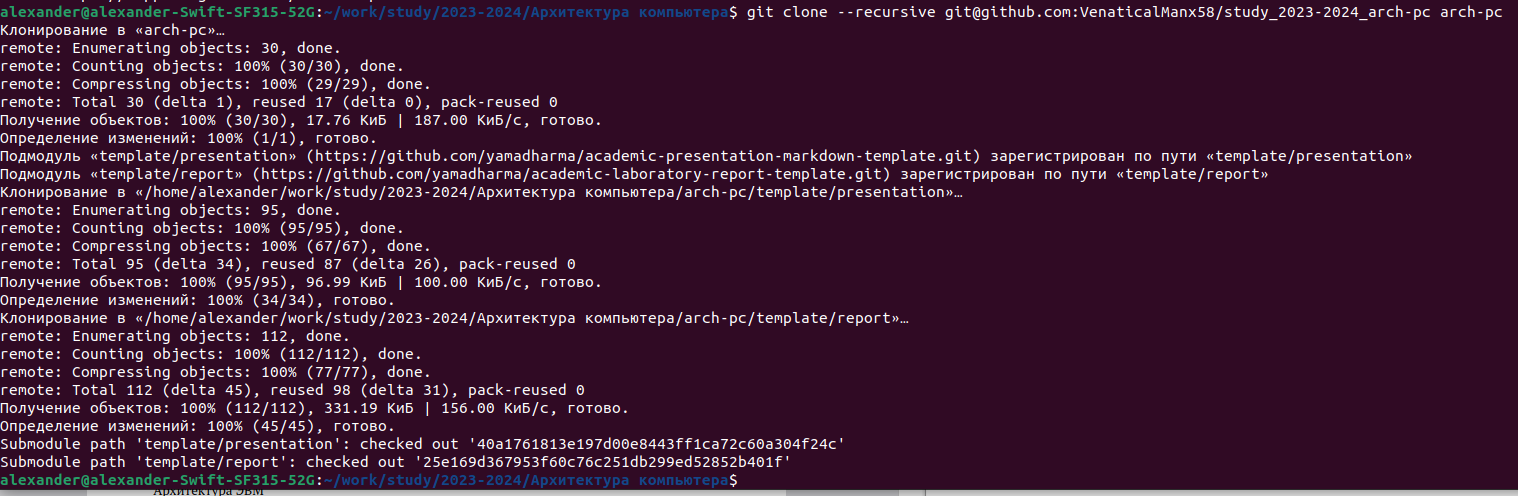


Рис. 8: Рисунок8

1. Настройка каталога курса Перейдите в каталог курса: cd /work/study/2023-2024/“Архитектура компьютера”/arch-pc Удалите лишние файлы: rm package.json Создайте необходимые каталоги: echo arch-pc > COURSE make Отправьте файлы на сервер: git add . git commit -am ‘feat(main): make course structure’ git push

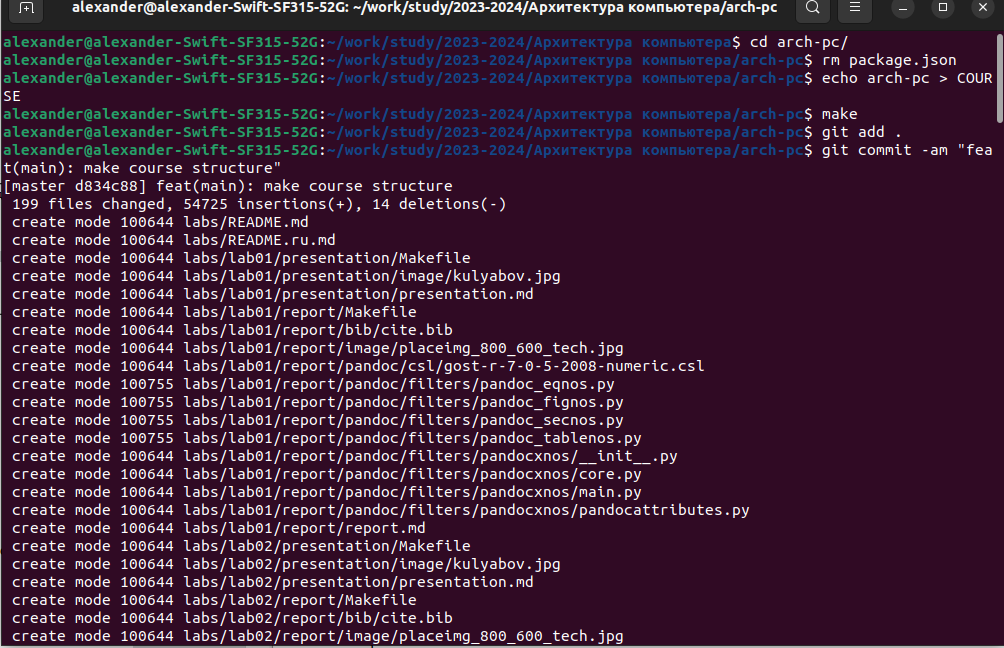


Рис. 9: Рисунок9

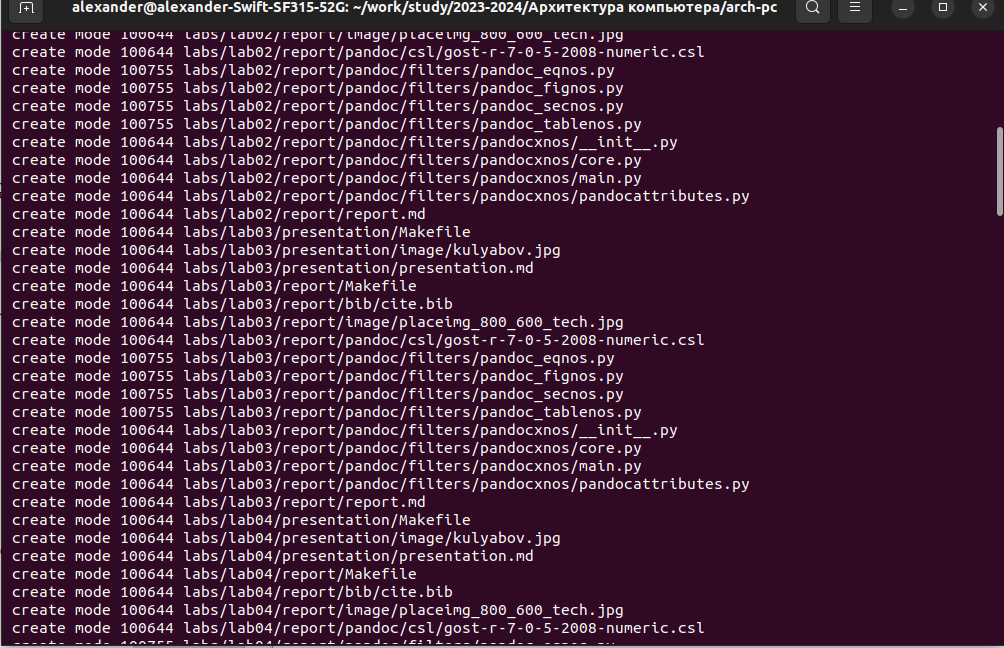


Рис. 10: Рисунок10

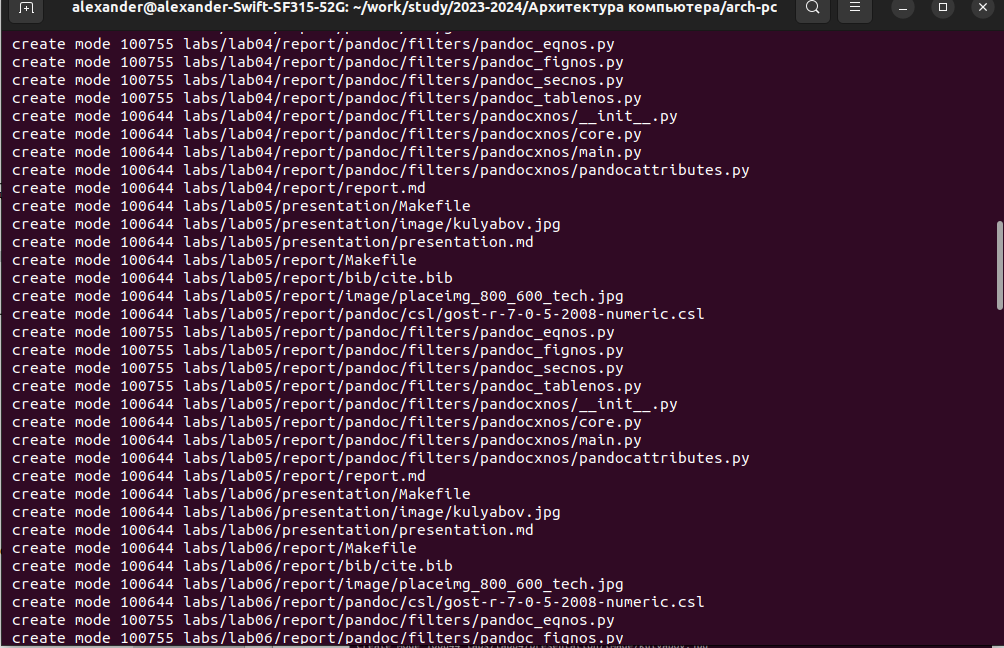


Рис. 11: Рисунок11

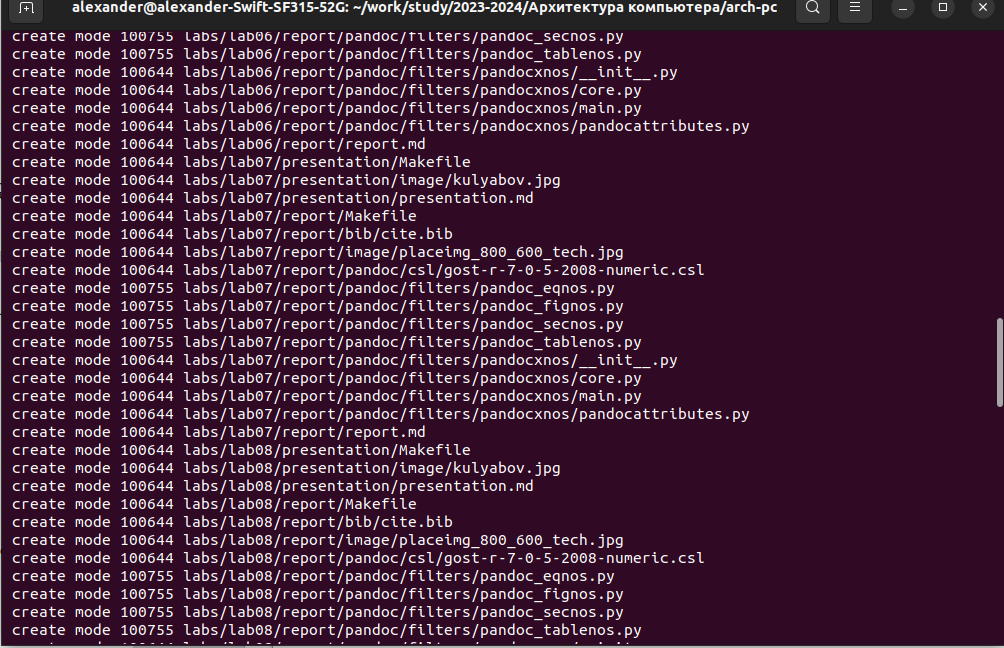


Рис. 12: Рисунок12

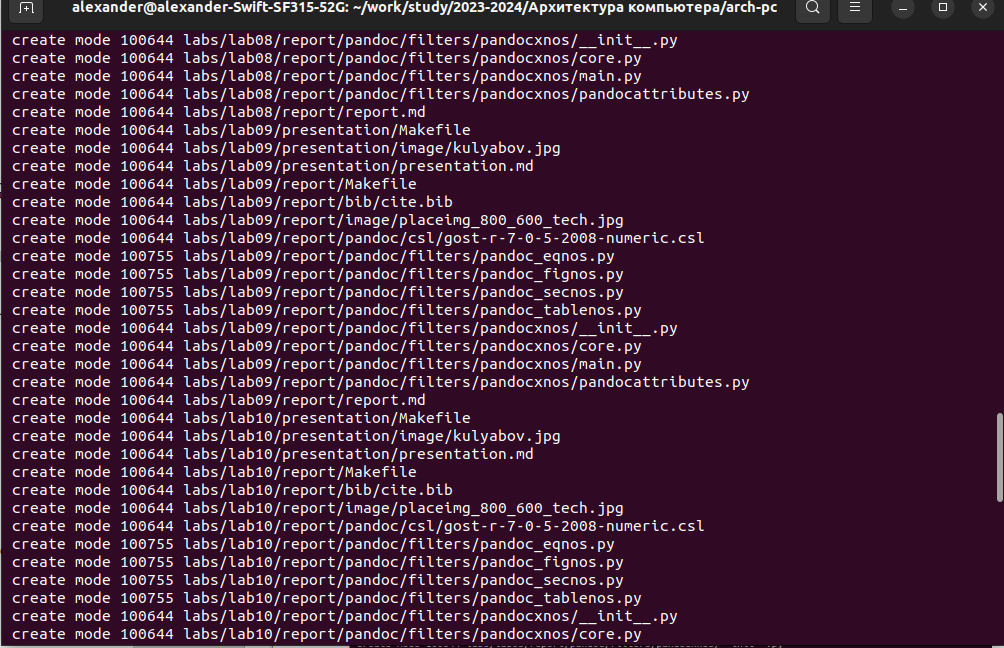


Рис. 13: Рисунок13

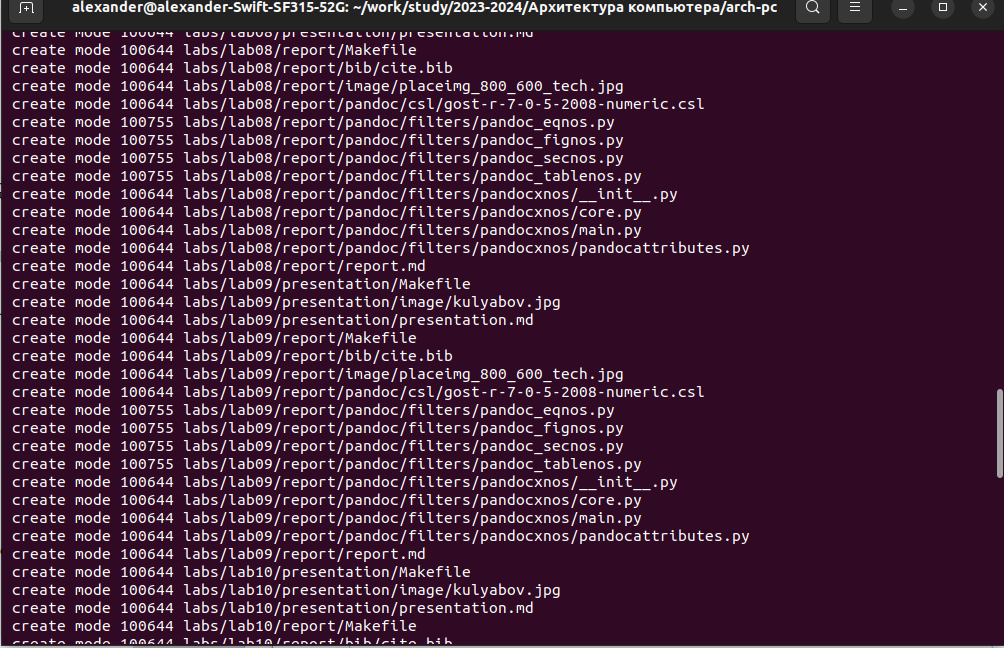


Рис. 14: Рисунок14

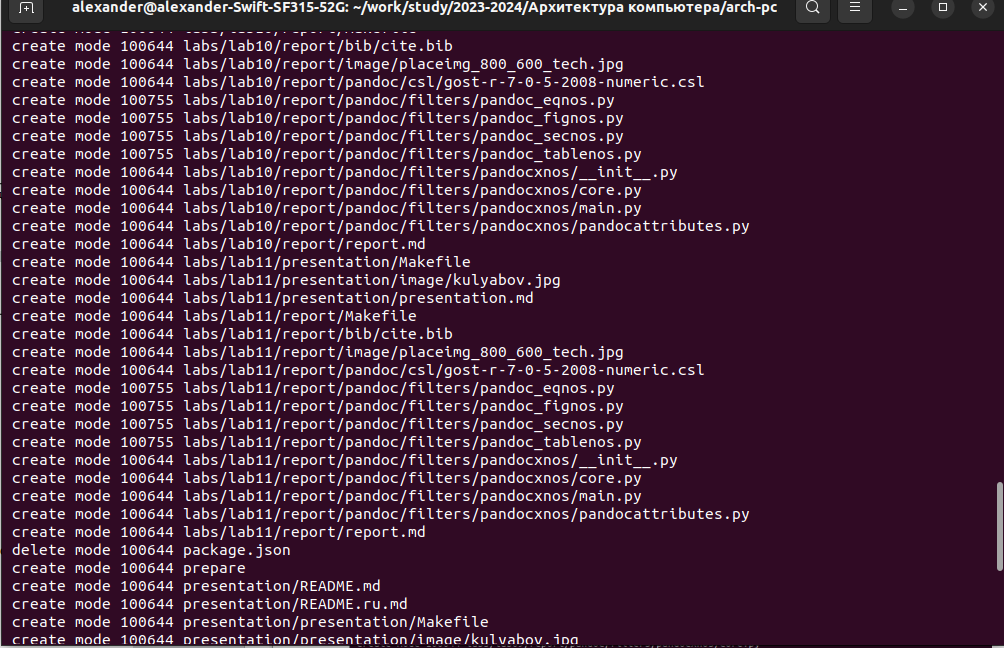


Рис. 15: Рисунок15

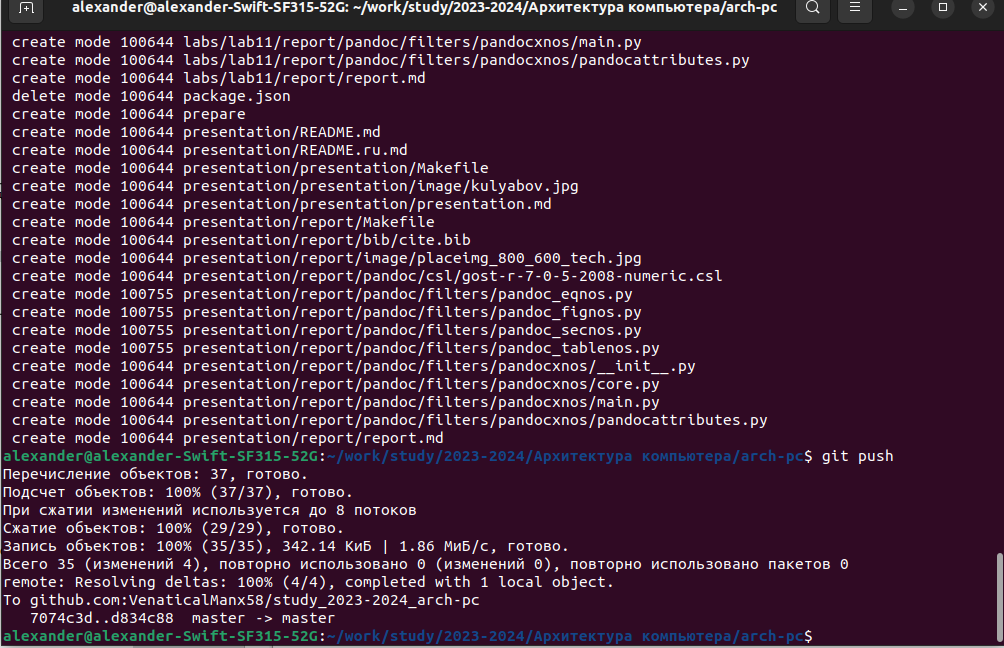


Рис. 16: Рисунок16

Проверьте правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

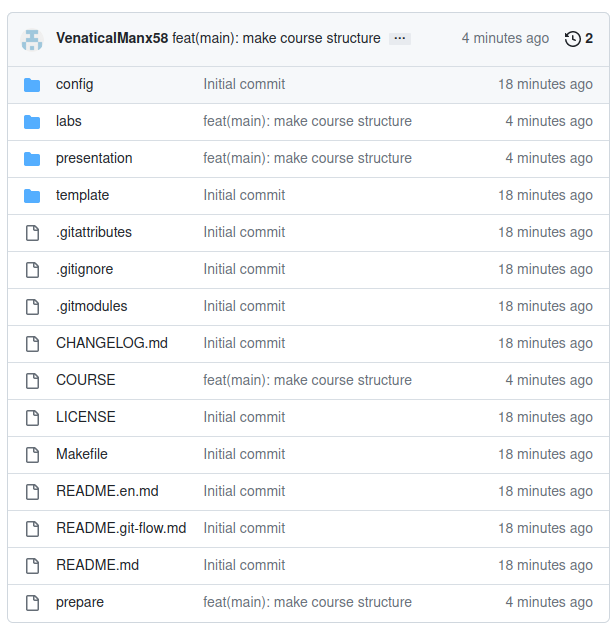


Рис. 17: Рисунок17

# 5 Задание для самостоятельной работы

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).

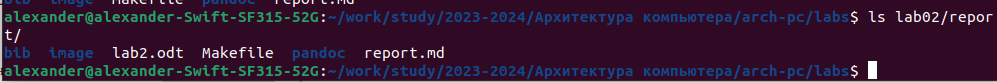


Рис. 18: Рисунок18

1. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

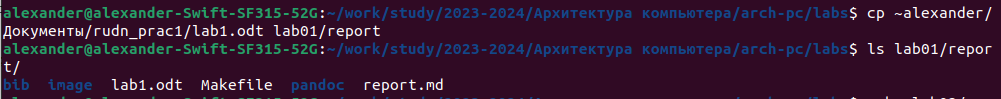


Рис. 19: Рисунок19

1. Загрузите файлы на github

# 6 Выводы

Мы научились пользоваться средством контроля версий git

# Список литературы