Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Сычев Егор Олегович

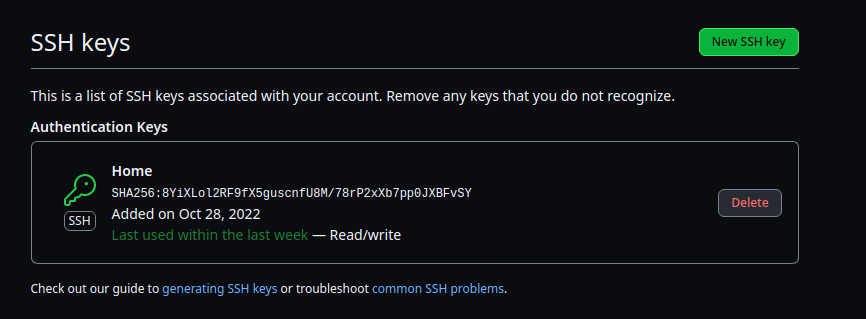
Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умение по работе с git.

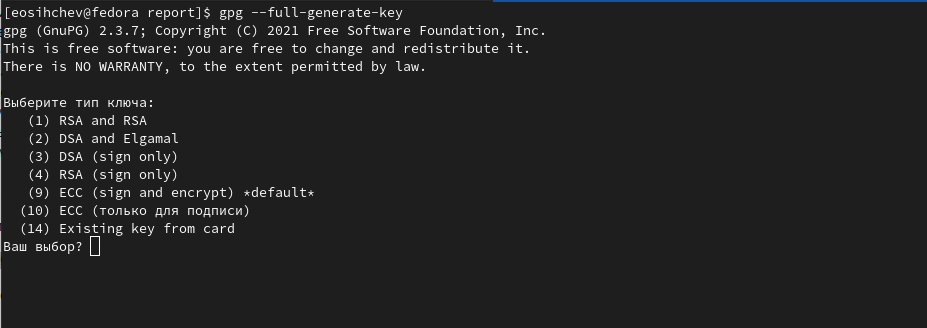
# 2 Выполнение лабораторной работы

1. Установка git,gh. Базовая настройка git. Создание ssh ключа. (Выполнено в предыдущих лабораторных работах)



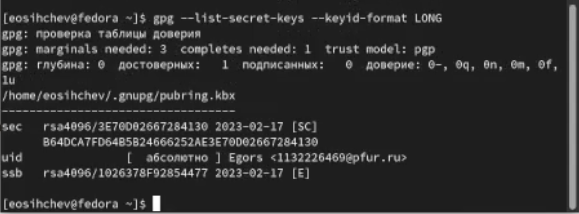
ssh github

1. Создание gpg ключа.



gpg –full-generate-key

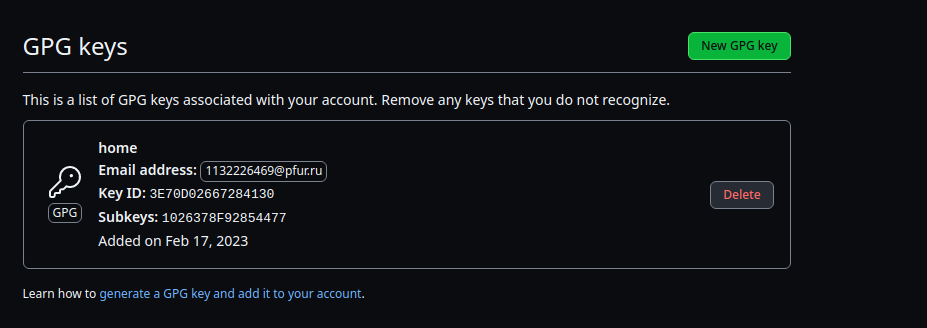
1. Добавление gpg ключа в github.



Копирование

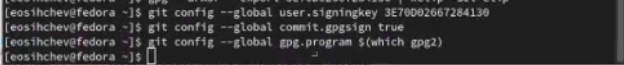
Копирование

Копирование



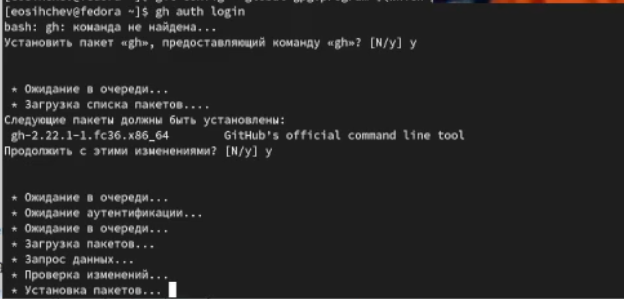
github gpg key

1. Настройка автоматических подписей коммитов git.

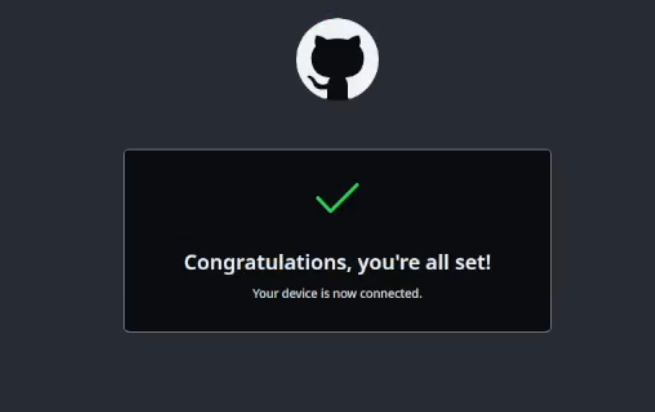


Настройка автомат. подписей

1. Настройка gh.

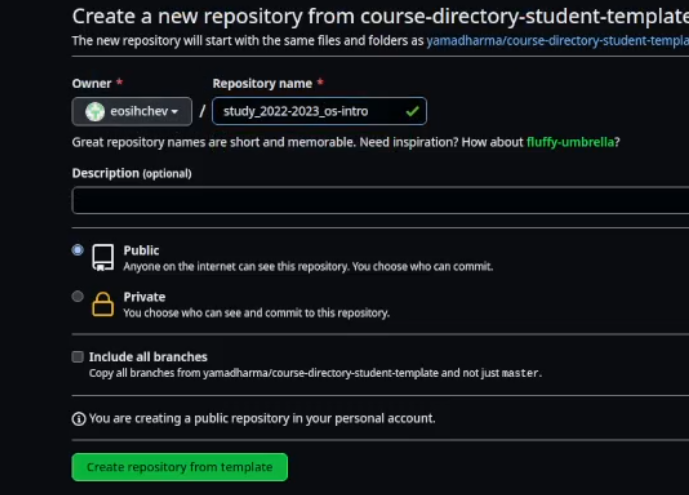


gh auth login

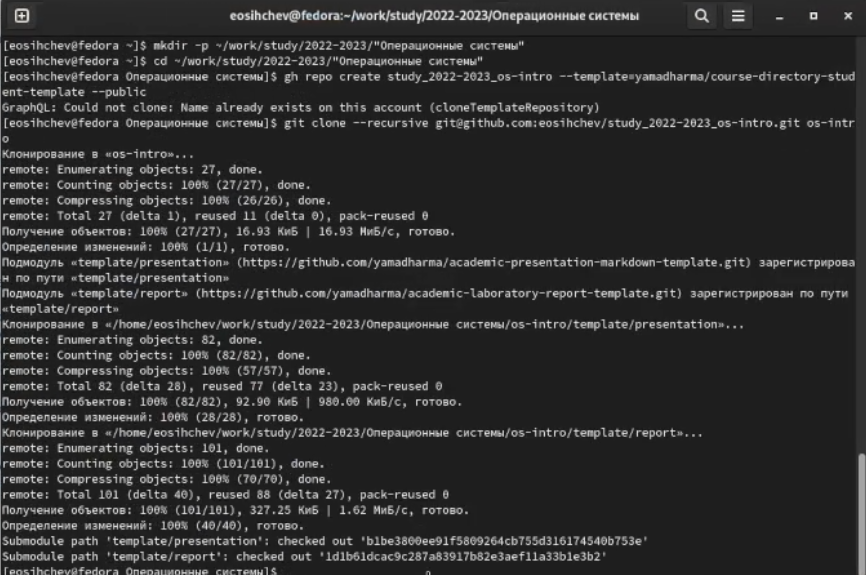


github авторизация

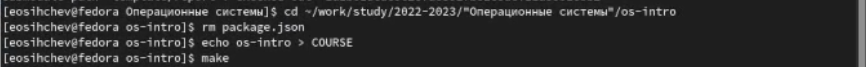
1. Создание репозитория курса на основае шаблона и настройка каталога курса.



Создание репозитория



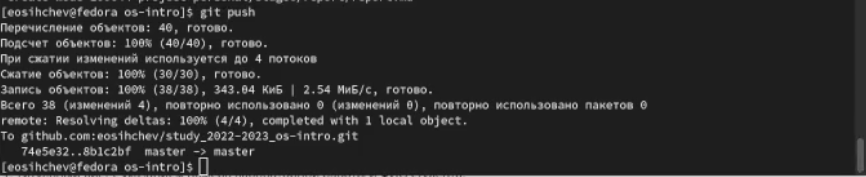
Клонирование



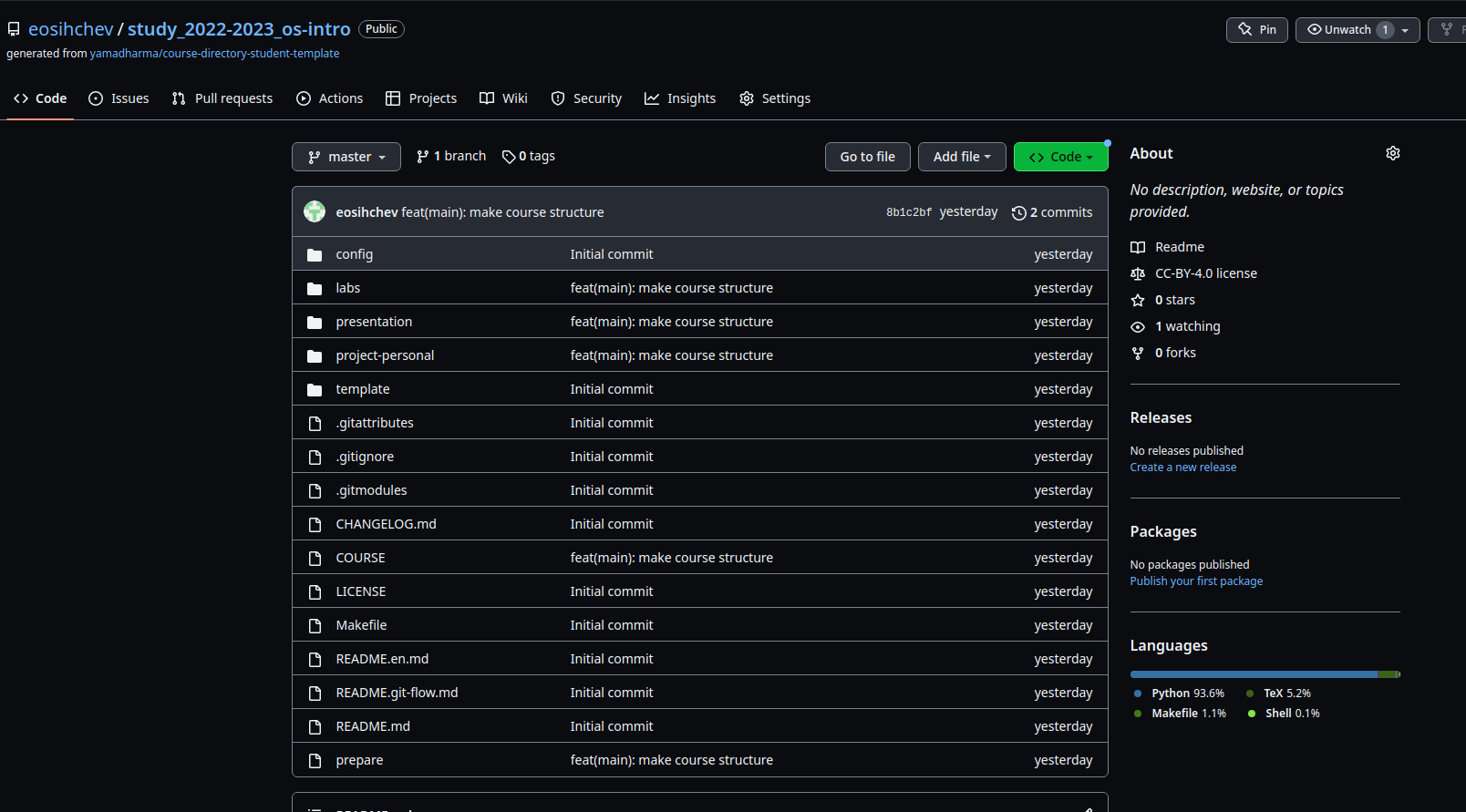
Настройка каталога

git add, git commit, git push

git add, git commit, git push



git add, git commit, git push



Получившийся репозиторий

# 3 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Контроль версий, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище (repository, сокр. repo), или репозитарий, — место хранения всех версий и служебной информации. • Коммит (commit; редко переводится как «слепок») — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»). • Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов.Выполнение большинства функций по управлению версиями осуществляется специальным сервером

1. Опишите действия с VCS при единоличной работе с хранилищем.

Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: git config –global user.name”Имя Фамилия” git config –global user.email”work@mail” и настроив utf-8 в выводе сообщений git: git config –global quotepath false Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке: сd mkdir tutorial cd tutorial git init

1. Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C”Имя Фамилия [work@mail](mailto:work@mail)” Ключи сохраняться в каталоге~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена сat ~/.ssh/id\_rsa.pub | xclip -sel clip вставляем ключ в появившееся на сайте поле.

1. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

1. Назовите и дайте краткую характеристику командам git.

Наиболее часто используемые команды git: – создание основного дерева репозитория: git init–получение обновлений (изменений)текущего дерева из центрального репозитория: git pull–отправка всех произведённых изменений локального дерева в центральный репози-торий: git push–просмотр списка изменённых файлов втекущей директории:git status–просмотртекущих изменения: git diff–сохранениетекущих изменений:–добавить все изменённые и/или созданные файлы и/или каталоги: git add .–добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов – удалить файл и/или каталог из индекса репозитория (приэтомфайл и/илик аталог остаётся в локальной директории): git rm имена\_файлов – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’–сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit–создание новой ветки, базирующейся натекущей: git checkout -b имя\_ветки–переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки–слияние ветки стекущим деревом: git merge –no-ff имя\_ветки–удаление ветки: – удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки–принудительное удаление локальной ветки: git branch -D имя\_ветки–удаление ветки с центрального репозитория: git push origin :имя\_ветки

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): git add hello.txt git commit -am’Новый файл

1. Что такое и зачем могут быть нужны ветви (branches)?

Ветки очень облегчают работу. Они решить такие проблемы как: нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко что-то напутать или потерять

1. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл. gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов: curl -L -s https://www.gitignore.io/api/list Затем скачать шаблон,например, для C и C++ curl -L -s https://www.gitignore.io/api/c >> .gitignore curl -L -s https://www.gitignore.io/api/c++ >> .gitignore

# 4 Вывод

Я изучил идеологию и применение средств контроля версий. И освоил умения по работе с git.