Лабораторная работа №2

Компьютерные науки и технология программирования. Операционные системы

Сячинова Ксения Ивановна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий git, а также освоить умения по работе с git.

2 Выполнение лабораторной работы

1. Для начала нам нужно создать учётную запись на github. Так как в прошлом семестре мы работали с данной платформой, то мы уже зарегестрировались там. (рис. 2.1).

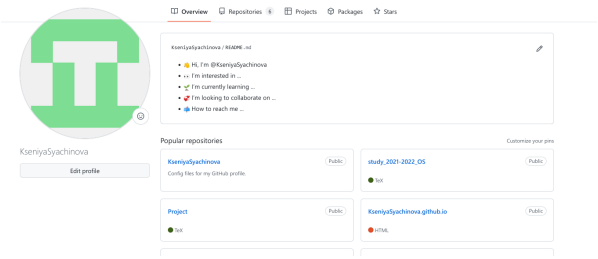

Рис. 2.1: Учётная запись

2. Затем произведём базовую настройку git, а именно:

- Зададим имя и email владельца репозитория
- Настроим utf-8 в выводе сообщений git
- Установка параметра autocrlf
- Установка параметра safecrlf (рис. 2.2).

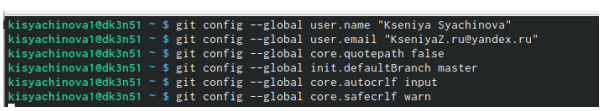

Рис. 2.2: Базовая настройка git

3. После этого создадим ключ для последующей индентификации пользователя на сервере. Делаем это с помощью команды “ssh-keygen -C Имя Фамилия ”.(рис. 2.3).

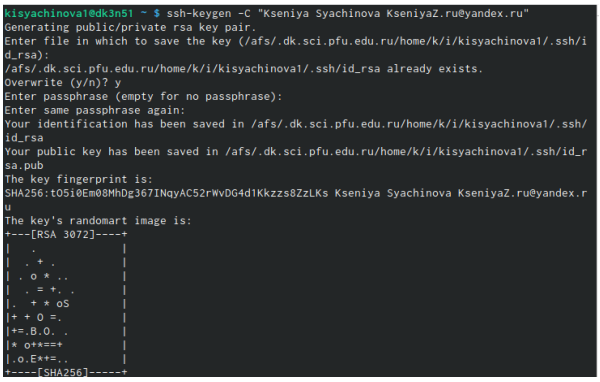

Рис. 2.3: Создание ключа

4. Командой “cat ~/.ssh/id_rsa.pub | xclip -sel clip” копируем этот ключ и загружаем его на github.(рис. 2.4)., (рис. 2.5).

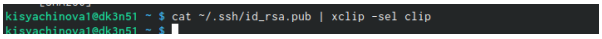

Рис. 2.4: Копирование ключа

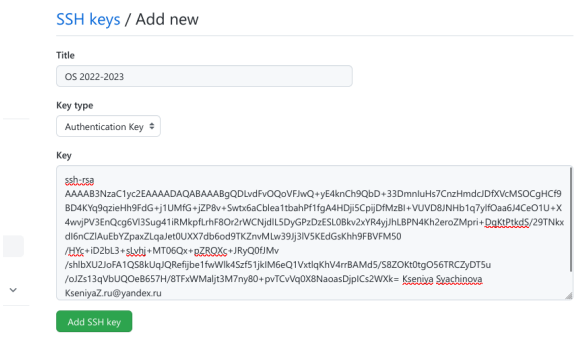

Рис. 2.5: Загрузка ключа

5. Далее создадим каталог “Операционные системы” в ранее созданном пространстве. Наш каталог будет иметь путь: work/study/2022-2023/“Операционные системы”.(рис. 2.6).

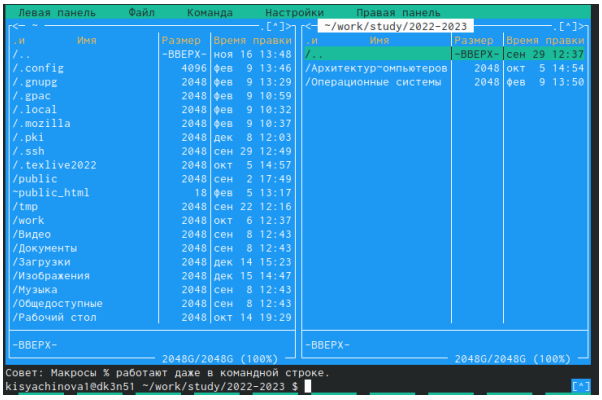

Рис. 2.6: Создание каталога

6. Создаём репозиторий, который будет создан на основе данного нам шаблона. (рис. 2.7), (рис. 2.8).

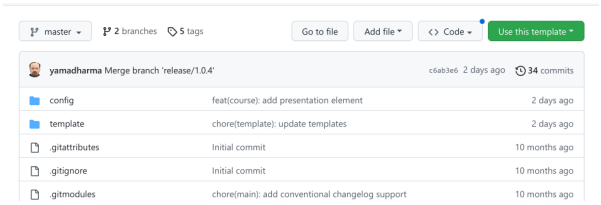

Рис. 2.7: Шаблон

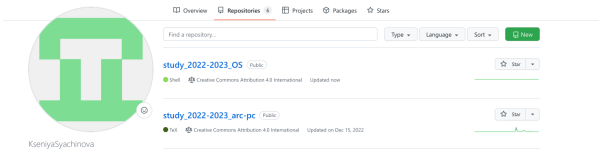

Рис. 2.8: Создание репозитория

7. Затем клонируем репозиторий. Для этогокопируем ссылку созданного репозитория (Code -> SSH) и с помощью команды “git clone –recursive” клонируем. (рис. 2.9),(рис. 2.10).

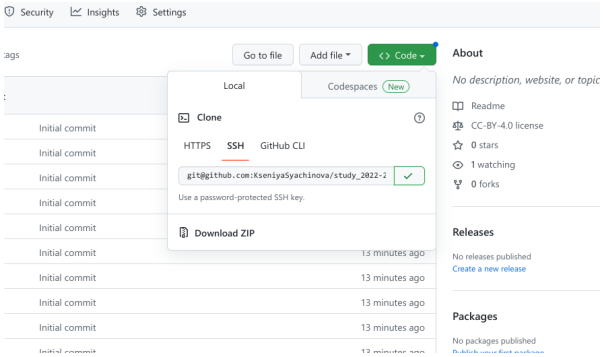

Рис. 2.9: Копирование кода

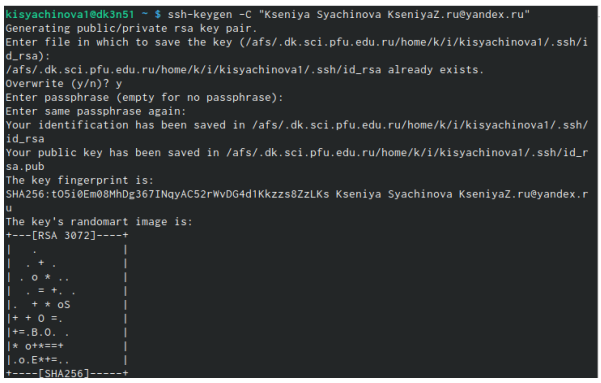

Рис. 2.10: Клонирвание репозитория

8. Настроим каталог курса, где удалим ненужные файлы, создадим необходимые каталоги и отправим файл на сервер.(рис. 2.11).

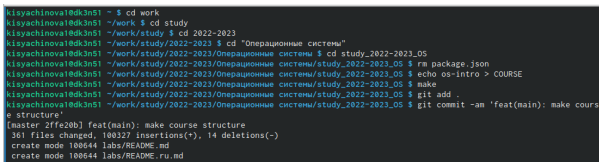

Рис. 2.11: Настройка каталога

9. Всё выполненно корректно. Мы создали рабочее пространство для выполнения дальнейших лабораторных работ. (рис. 2.12).

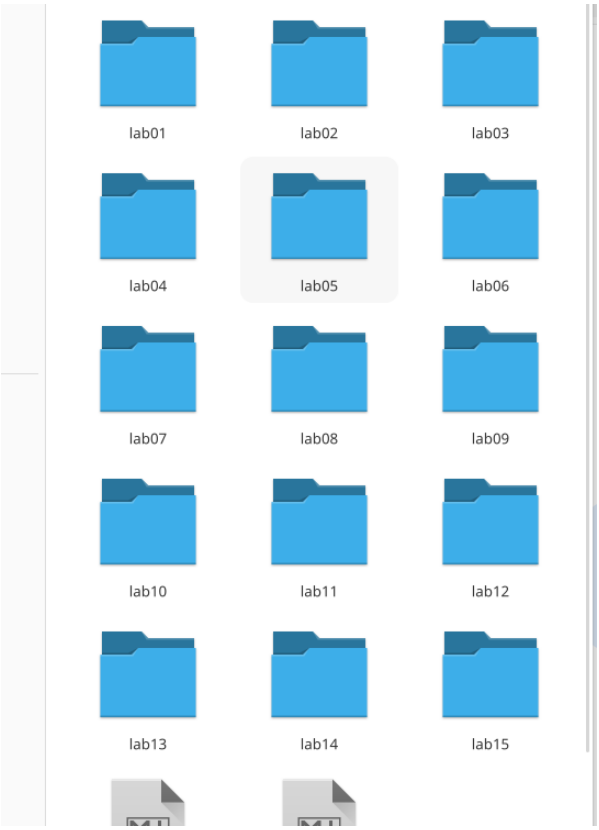

Рис. 2.12: Рабочее пространство

3 Выводы

В процессе выполнения данной лабораторной работы я создала рабочее пространство для дальнейшей работы. Так же вспомнила идеологию и средства контроля версий git.

4 Ответы на контрольные вопросы

1. 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. Какие задачи решает система контроля версий:

- Защищает исходный код от потери. Данные хранятся на удалённом сервере, даже если разработчики удалят файлы с локального компьютера, они останутся в репозитории.
- Обеспечивает командную работу.
- Помогает отменить изменения.
- Распределённая работа.

2.

- 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)

3. 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

4. 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:
cd
mkdir tutorial
cd tutorial
git init

5. Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C”Имя Фамилия work@mail” Ключи сохраняться в каталоге~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена

cat ~/.ssh/id_rsa.pub | xclip -sel clip
вставляем ключ в появившееся на сайте поле.

6. 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.

7. 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

8. Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

git add hello.txt
git commit -am’Новый файл

9. 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

10. 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