

Лабораторная работа №2

**Компьютерные науки и технология программирования. Операционные
системы**

Сячинова Ксения Ивановна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11
4	Ответы на контрольные вопросы	12

Список иллюстраций

2.1	Учётная запись	6
2.2	Базовая настройка git	6
2.3	Создание ключа	7
2.4	Копирование ключа	7
2.5	Загрузка ключа	7
2.6	Создание каталога	8
2.7	Шаблон	8
2.8	Создание репозитория	8
2.9	Копирование кода	9
2.10	Клонирование репозитория	9
2.11	Настройка каталога	9
2.12	Рабочее пространство	10

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 зарегистрировались там. (рис. 2.1).

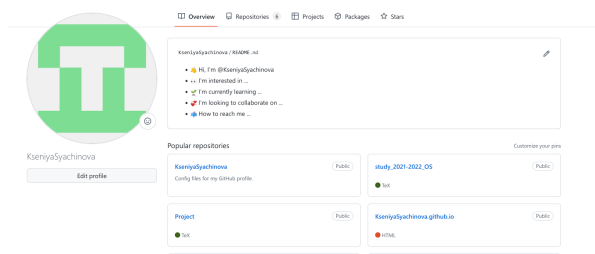


Рис. 2.1: Учётная запись

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).

```
kisyachinova@dk3n51 ~ $ git config --global user.name "Kseniya Syachinova"
kisyachinova@dk3n51 ~ $ git config --global user.email "Kseniya2.ru@yandex.ru"
kisyachinova@dk3n51 ~ $ git config --global core.quotepath false
kisyachinova@dk3n51 ~ $ git config --global init.defaultBranch master
kisyachinova@dk3n51 ~ $ git config --global core.autocrlf input
kisyachinova@dk3n51 ~ $ git config --global core.safecrlf warn
```

Рис. 2.2: Базовая настройка git

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

```
kisyachinova1@dk3n51 ~ $ ssh-keygen -C "Kseniya Syachinova KseniyaZ.ru@yandex.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:t05i0Em08MhDg367INayAC52rWvDg4d1Kkzss8ZzLKs Kseniya Syachinova KseniyaZ.ru@yandex.ru
The key's randomart image is:
+---[RSA 3072]-----+
|
| . + .
| . o + .
| . = + .
| . + oS
|+ + 0 =
|+=B.O.
|X o+==
|..O.E+==
+---[SHA256]-----+
```

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).

```
kisyachinova1@dk3n51 ~ $ cat ~/.ssh/id_rsa.pub | xclip -sel clip
kisyachinova1@dk3n51 ~ $
```

Рис. 2.4: Копирование ключа

SSH keys / Add new

Title
OS 2022-2023

Key type
Authentication Key

Key
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDvdFvOQoVfhwQ+yE4knCh9QbD+33DmnlHs7CnzHmdcIDfVxVMSOCgHCP9
BD4Kvq9pzieHf9dG+j1UMfG+j2P8v+Swtx6aCblea1tbahP1fGa4HDj5CpjdPMz8li+VUVDB/NHb1q7YfIOaa6I4CceO1U+X
4wvPV3EnQc6V13Sug41RMkplfHf8Or2WCNjdLSDyGpZdZESL0Bkv2YR4yJhLBPNA4kh2ero2Mpri+DqKPkds/29TNKx
d6nCZlAuEbyZpaxZLqaJel0UXX7db6od9TKZmvMlw39j3V5KEdGskNH9FBVFM50
/hXc+iD2bL3+slxj+MT06Qx+gZ8QKc+JRYQ0fIMv
/shbXU2roFA1Q58kUqIQRefjbe1hwWlk4Szf51jklM6eQ1VxtlqKhV4rBAMd5/S8ZOKt0tgO56TRCZyDT5u
/oiZs13qVbUQOeB657H/8TFxWMajl3M7ny80+pvTCvQdX8NaoasDjplCsz2Wxk+ Kseniya Syachinova
KseniyaZ.ru@yandex.ru

Add SSH key

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).

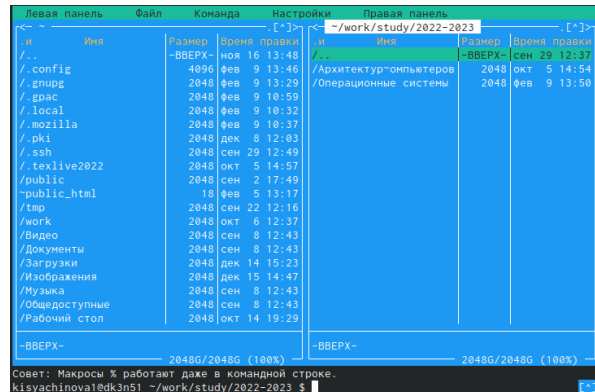


Рис. 2.6: Создание каталога

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

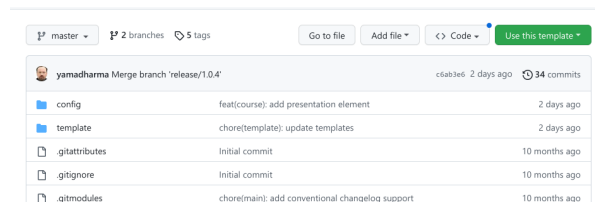


Рис. 2.7: Шаблон

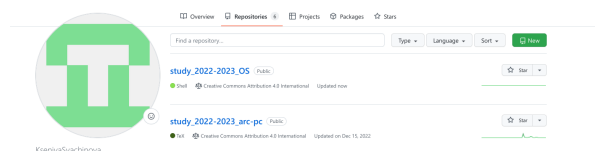


Рис. 2.8: Создание репозитория

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

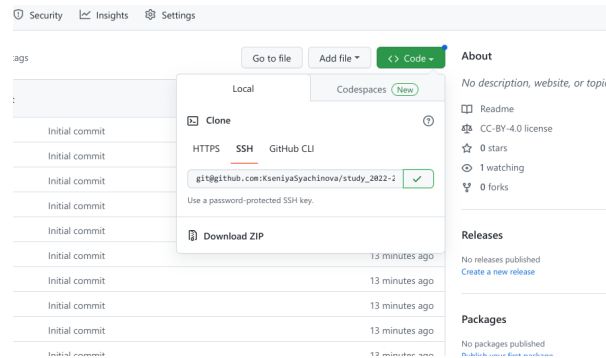


Рис. 2.9: Копирование кода

```
kisyachinova@k3n51 ~ $ ssh-keygen -C "Kseniya Syachinova KseniyaZ.ru@yandex.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:t05i0Em0MhDg367INqyAC52rWvDg4d1Kkzzs8ZzLKs Kseniya Syachinova KseniyaZ.ru@yandex.ru
The key's randomart image is:
+-----[RSA 3072]-----+
| . + |
| . O * . |
| . + * . |
| + * O S |
| + + O = |
| = B O . |
| = O * * + |
| O E * + . |
+-----[SHA256]-----+
```

Рис. 2.10: Клонирование репозитория

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

```
kisyachinova@k3n51 ~ $ cd work
kisyachinova@k3n51 ~/work $ cd study
kisyachinova@k3n51 ~/work/study $ cd 2022-2023
kisyachinova@k3n51 ~/work/study/2022-2023 $ cd "Операционные системы"
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы $ cd study_2022-2023_05
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы/study_2022-2023_05 $ rm package.json
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы/study_2022-2023_05 $ echo os-intro > COURSE
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы/study_2022-2023_05 $ make
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы/study_2022-2023_05 $ git add .
kisyachinova@k3n51 ~/work/study/2022-2023/Операционные системы/study_2022-2023_05 $ git commit -am 'feat(main): make course structure'
[master 2ffe20b] feat(main): make course structure
261 files changed, 10022 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
```

Рис. 2.11: Настройка каталога

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

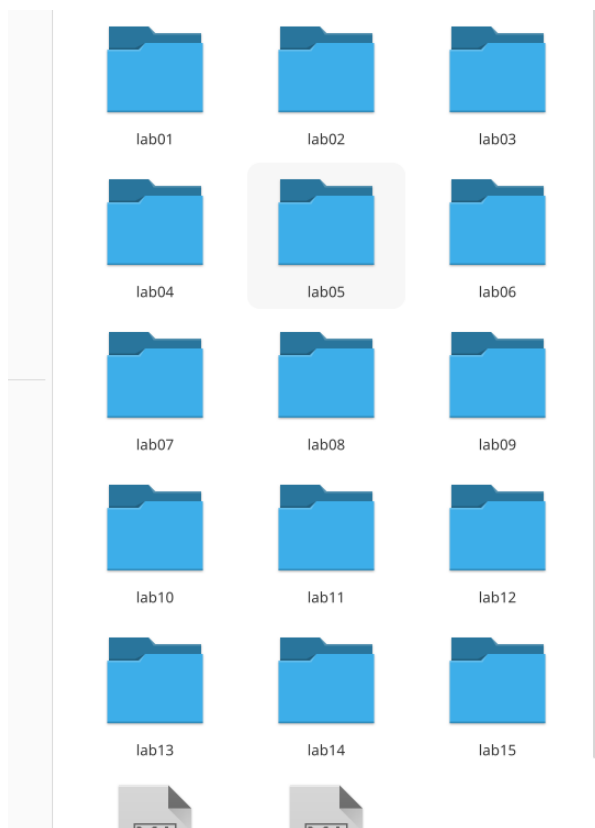


Рис. 2.12: Рабочее пространство

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 репозиторий, — 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 core.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 репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются 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 репозиторий:
- `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/иликаталог 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 сли-
той 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. Использование `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, со-
здаваемые компиляторами. Можно прописать шаблоны игнорируемых при
добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сер-
висов. 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
```