

# **Отчёт по лабораторной работе №2**

**Операционные системы**

Ничипорова Елена Дмитриевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выводы	14

## Список иллюстраций

3.1	Синхронизация . . . . .	8
3.2	Настройка системы работы с версиями git . . . . .	9
3.3	ключ ssh . . . . .	9
3.4	Создание репозитория . . . . .	10
3.5	grpg ключ . . . . .	11
3.6	grpg ключ в гитхабе . . . . .	11
3.7	создание шаблона рабочего пространства . . . . .	11
3.8	созданный репозиторий . . . . .	12
3.9	создание нового репозитория . . . . .	12
3.10	созданный репозиторий . . . . .	13

## Список таблиц

# 1 Цель работы

изучить идеологию и применение средств контроля версий, освоить умения по работе с git.

## 2 Задание

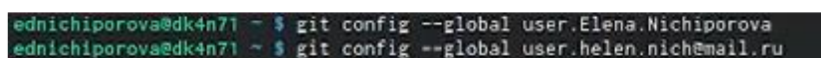
Создать базовую конфигурацию для работы с git. – Создать ключ SSH. – Создать ключ PGP. – Настроить подписи git. – Зарегистрироваться на Github. – Создать локальный каталог для выполнения заданий по предмету

### 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 зави-

симости 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. # Выполнение лабораторной работы

Изначально я зарегистрировалась на github. Затем синхронизируем учетную запись с компьютером через консоль (рис.1) (рис. 3.1)

A screenshot of a terminal window showing two lines of commands being executed. The first line is 'ednichiporova@dk4n71 ~ \$ git config --global user.Elena.Nichiporova' and the second line is 'ednichiporova@dk4n71 ~ \$ git config --global user.helen.nich@mail.ru'. The prompt character is a green dollar sign, and the output is not visible, suggesting the commands are still being processed or the screenshot was taken before completion.

```
ednichiporova@dk4n71 ~ $ git config --global user.Elena.Nichiporova
ednichiporova@dk4n71 ~ $ git config --global user.helen.nich@mail.ru
```

Рис. 3.1: Синхронизация

Дальше настраиваем систему работы версий с git. После этого создаем ключ ssh и копируем его через консоль (рис.2)(рис. 3.2)



```
ednichiporova@dk4n71 ~ $ ssh-keygen -C"Elena.nichiporova<helen.nich@mail.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/e/d/ednichiporova/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/d/ednichiporova/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/d/ednichiporova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:6j4zqLn/SSVUoQL01IvSL6Mz7G6vCovxF6E9cmly10 Elena.nichiporova<helen.nich@mail.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|
| o o +
| . . * o E
| . . o = S
| . o =
| .ooo. =
|<=+. +
|Xoo+oo+
|O#X=o+++
+----[SHA256]-----+
ednichiporova@dk4n71 ~ $ cat ~/.ssh/id_rsa.pub | xclip -sel clip
ednichiporova@dk4n71 ~ $
```

Рис. 3.2: Настройка системы работы с версиями git

Далее копируем его в github (рис.3)(рис. 3.3)

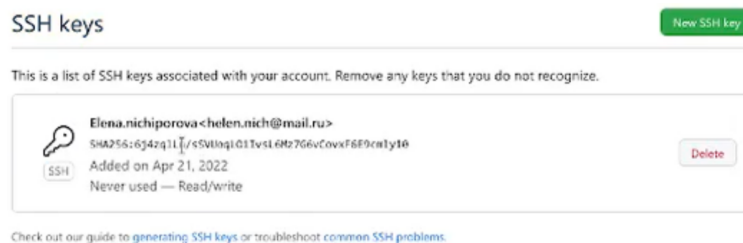
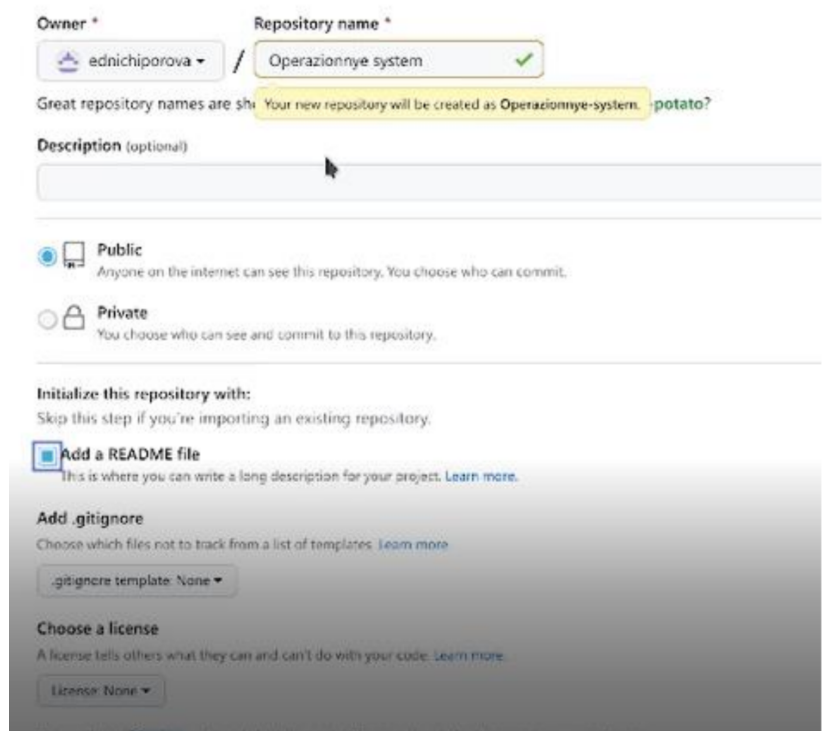


Рис. 3.3: ключ ssh

Дальше мы создаем репозиторий в github. После этого копируем ссылку на него и благодаря этому мы сможем работать с его папками и файлами через консоль (рис.4)(рис. 3.4)



Owner \* / Repository name \*

ednichiporova / Operazionnye system ✓

Great repository names are short, lowercase, and contain only numbers, letters, hyphens, and underscores. Your new repository will be created as Operazionnye-system. potato?

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)  
.gitignore template: None ▼

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)  
License: None ▼

Рис. 3.4: Создание репозитория

Далее создаем ключ pgr с помощью команды `gpg --full-generate-key`, копируем его через консоль (рис.5)(рис. 3.5)

```

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINB6JhRiABEADqf1KnPg7bNqLTC1DnUNNwR52vYDhebIhI K7cR9e14qs0/e
Z5fyvY1b1nef XGRHMLC5wtY69126xwH4ulwF EhwumvFgheSjx0vt4ZLEDHppm/
oYdrqZ2JXQ6KjnzUB10hts1Z59y2AHd4Gu1wF7FY1p8x9pJqhehIXZMIKfACEGD
q1kshv/BDTBC0Z7EmdcsVJ7S+Df27U0XmwS1418TMcpIer9uJd/02cV/yYg7HGGL
fIKSvH5gplXpCzSZv4RZJ1U0qOZYrfl04osjbQJ7efCF8ejbVAsAwKAbuHwHxHG
WuPLNCDaJ1+q+w+PyMtOrwAv+WsPrQ4GjYuy/0LQ+PjN3KcGL3aRQbH4frru8Fi
cKX9VB5vbe0iEXoMYfWJQv1ALC6Wry/GPY2PRf/qnoyY4AIdD59AD9Y10Dcejqm
Pl/PXwZ0EAr/AzeXBS05Cik/wbaqZYU2ZuZuZbHwYAOVXpYhKwdk1+QcYNz51m6
ZbMbUfCwSAtfC82nyuB9fCCfHkQUJ7f/EnXDURx3BW6+ned4wneZ5vY3e3yuMQ5
i.95R8KipD6mTpyw0l0fzG6XynzfgUaUACFm+BFpLkJKRCWtVbH5wVJM5hI11
sN10Jq9ifUkKf4+tfvub06fr1YcUn0U0fn+baUxn1U30c4V1+3rNAj3bWARAQAU
tBpF6WUyV5A8aGVsZM4ubmljaEBTYW1sLnJ1PokTcJtAQwQAC0BvBHf5yptJ329m
PAZ61ooi7h1XBS0BQJ1YUyGfAhsDBQsJCACeRHMLCKQJLGAQWgMBAfA8AAhAAoA
fEl0izhLXUosQbe4QALCkCM6e7g1onn8YV10S0C42Fxcw7rDg+blM1fYCW1v91f
7qWio3KJ9fJtNAUI+txKsph5UwtJGtZaoNvXpIknq+vg7o9z1uypN1ns56T1aQD
36857fJQlCC1fFKz71aipd6enhsoRmSn2RozJWpUr+v67JmBpTS0gehlvfqL
HJ0tSL7fYp1Kf1UPihabLIR10RuZ1w8ZGM15wiXugU1Zgl2Jveq/oIeWJf/0+Dbl
4CQ+Zdobf5DRNhhMozcrJBMJbbXbCKf1fVZv1i0MsjEMCJT1Y7b7d749WvXp
wydIt1XETStjtdKaiSaTeoVWd41RBKz7s65Qd0pbhI3hjp+8GEXgr+MBT1UwkwJl
R+19Uyq70YNI155K/nm3/EbLhU1jSfJRE1PncZCVvrnoM0s+eayT5Qgub6L04t5
ca5J7Y6Zf8hnTPpRBR7UB/JalIKZB456hJj1ISZWV1A1I71cmY3Qus6m60KEs
FKF61v58u0D9/iscc1DxdT+zwQjsg56hkdJ0nPnaJXgR3R1QVNZ3y2Mhm0a2x0TLf
VJgJx3bJ9i0/RUd+IRRCZ5Jw1J2Et0XLt05+eKyU5MKY4f1RhZSn2kznz+qYryHh
G69957EDCnBJRv05JfPqQOCshK+AzJai1ct8HfAZ9v+3wguJ27PIHUS3/C4uQIN
BGJhRiABEADvY12+HAOfkK4yMMcYEv9v9r0B6K6d3Zf0I7G1w/NalimKJkch
mlrVpYbHwXsnnXWMI8327db+oumKwMlthras3sw/NKSCeQWpY7mL/SaC5mp9h
xiurmuYbFqA29r2DEJLw7Xk3l7xtibmf1mYrfvx0ntne60u0FVFqYh5eMjBkIgK
bg9NFKhlpw4/7X2X/x50eHFETjY36r39r4V7Hui/Nm0JDMiQfYMKH8GKtKsZc
zbn2qkhsYfM5/DnzUwAmYXncL56eSXK1Uuq4gfZuU7bjbKoyAwnjpcZLZcZc
olnXSEMBDn9vA10gV0S0E5104KlY18jEYQNFxeFFPOyAtBJfYingln4wAYJJK
/MI9p0QNZ7qJt9I9AAjddRAX0ur+H8W7oxk4iQxGNgfXoqM0fDcAMJ0f/0oqHh
mH3M1fyG7ajtuwko1XnB61Y1JkxUf2QUUTNI/MfJE1x/1f4k1paqZLuu31/GH

```

Рис. 3.5: gpg ключ

Переходим в настройки github и вставляем полученный ключ(рис.6)(рис. 3.6)

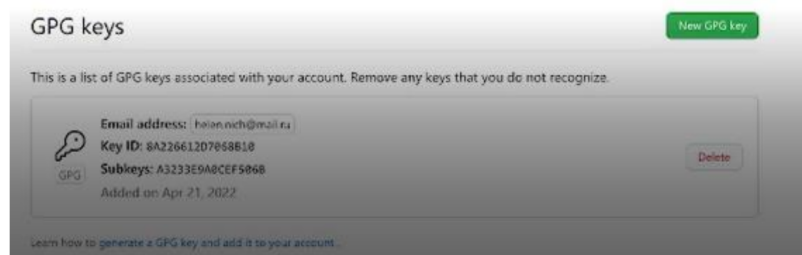


Рис. 3.6: gpg ключ в гитхабе

Дальше создаем репозиторий курса на основе шаблона. Для этого создаем шаблон рабочего пространства с помощью команд (рис.7)(рис. 3.7)

```
mkdir -p ~/work/study/2021-2022/"Операционные системы"  
cd ~/work/study/2021-2022/"Операционные системы"
```

Рис. 3.7: создание шаблона рабочего пространства

Используем уже созданный репозиторий (рис.8)(рис. 3.8)

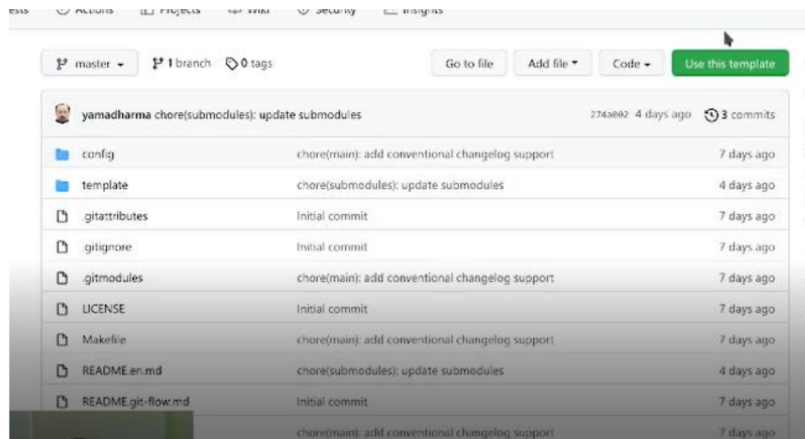


Рис. 3.8: созданный репозиторий

Создаем новый репозиторий(рис.9)(рис. 3.9)

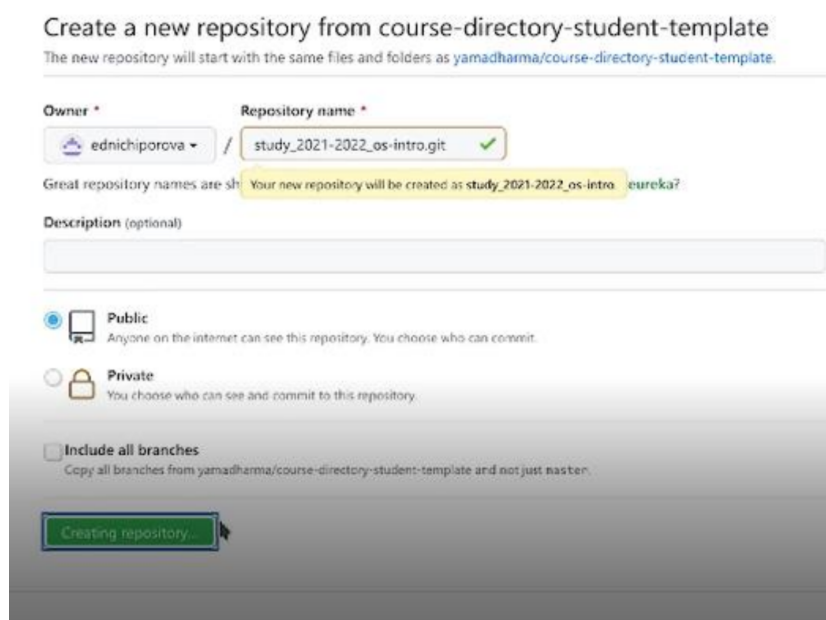


Рис. 3.9: создание нового репозитория

Далее настраиваем каталог курса , то есть удаляем ненужные файлы, создаем необходимые каталоги и отправляем файлы на сервер (рис.10).(рис. 3.10)

```
ps-intro $ makeCOURSE=os-intro
ps-intro $ git add .
ps-intro $ git add .
ps-intro $ git commit -am'feat(main): make course structure'
ps-intro $ git push
```

Рис. 3.10: созданный репозиторий

## 4 Выводы

В данной лабораторной работе я научилась работать с Github (создавать и привязывать учетную запись к компьютеру). Разобрала основные команды git и рассмотрела как их применять их при работе с Github. Изучила идеологию и научилась применять средства контроля версий.