

# **Отчёт по лабораторной работе №2**

**Операционные системы**

Бекауов Артур Тимурович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>14</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>15</b>

## Список иллюстраций

3.1	Установка пакетов git . . . . .	7
3.2	Базовая настройка git . . . . .	7
3.3	Создание pgr ключа . . . . .	8
3.4	Вывод списка ключей . . . . .	8
3.5	Копирование ключа . . . . .	9
3.6	Добавление GPG ключа в GitHub . . . . .	9
3.7	Автоматические подписи коммитов . . . . .	9
3.8	Авторизация устройства в GH . . . . .	10
3.9	Создание репозитория по шаблону . . . . .	10
3.10	Создание каталога для репозитория . . . . .	11
3.11	Создание ssh ключей . . . . .	11
3.12	Добавление ssh ключа . . . . .	12
3.13	Копирование репозитория курса . . . . .	12
3.14	Отправка изменений на сервер . . . . .	13

## Список таблиц

# 1 Цель работы

Цель данной лабораторной работы – изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

## 2 Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Зарегистрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

### 3 Выполнение лабораторной работы

Первым делом в начале лабораторной работы я проверил, что у меня установлены пакеты системы контроля версий git. (рис. 3.1).

```
[atbekauov@atbekauov ~]$ sudo dnf install git

Последняя проверка окончания срока действия метаданных: 0:30:44 назад, Пт 21 июн 2024
11:02:52.
Пакет git-2.45.2-2.fc40.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка пакетов git

Далее задаю имя и email владельца репозитория, Настраиваю utf-8 в выводе сообщений git, задаю имя начальной ветки и указываю параметры autocrlf и safecrlf (рис. 3.2).

```
[atbekauov@atbekauov ~]$ git config --global user.name "Artur Bekauov"
[atbekauov@atbekauov ~]$ git config --global user.email "Artbeka@yandex.ru"
[atbekauov@atbekauov ~]$ git config --global core.quotepath false
[atbekauov@atbekauov ~]$ git config --global init.defaultBranch master
[atbekauov@atbekauov ~]$ git config --global core.autocrlf input
[atbekauov@atbekauov ~]$ git config --global core.safecrlf warn
[atbekauov@atbekauov ~]$
```

Рис. 3.2: Базовая настройка git

Затем создаю ргр ключ, по указанным в методчке параметрам. (рис. 3.3).

```

gpg: /home/atbekauov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/atbekauov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/atbekauov/.gnupg/openpgp-revocs.d/A7632A7EB28E00762410FF5B92A776B92CF55346.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-06-21 [SC]
      A7632A7EB28E00762410FF5B92A776B92CF55346
uid           Artur Bekauov <Artbeka@yandex.ru>
sub   rsa4096 2024-06-21 [E]

[atbekauov@atbekauov ~]$ █

```

Рис. 3.3: Создание pgr ключа

После этого вывожу список ключей и копирую отпечаток приватного ключа.  
(рис. 3.4).

```

[atbekauov@atbekauov ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных: 1  подписанных: 0  доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/92A776B92CF55346 2024-06-21 [SC]
      A7632A7EB28E00762410FF5B92A776B92CF55346
uid           [ абсолютно ] Artur Bekauov <Artbeka@yandex.ru>
ssb   rsa4096/7285D7A08FF486A3 2024-06-21 [E]

[atbekauov@atbekauov ~]$ █

```

Рис. 3.4: Вывод списка ключей

У меня не корректно работал xclip, поэтому вывожу ключ в терминале, и копирую его вручную в буфер обмена. (рис. 3.5).



```
[atbekauov@atbekauov ~]$ gpg --armor --export 92A776B92CF55346
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGZ1PQYBEADH0K7xDLK2aXfY5rXP46CkY96+z3fmoYCPFOgm9mbFbkad5mAy
hsZjGC+Xk+OeBKDJ360bnn160ytZHxipkgr2oKEpc1TIyaqu9mWsrigh1fNjjnDO
B6UwOZXNxx9MfPcf9NzooE2BeyRA50B3ued+qEpqmWd51DfUnCR5eF0r+ssZiAwO
sE1JCajWhNNhYyhtdWfuV6txg6jB/DLSBmWhtN2f8c7X9Fy5mj1r2eYXpzp77lex
sV6heQC67wDXimjdfRhwGwGEf8kqN4CgHeczp490LU5R9ymDxzjT028Mw04pIzap
vepV3qyPf1ge535BTSWro0fYwaVBWwQNaOwheBCyCySut1zUeXP2RN0Xu8VGqk+q
usNE0CZqbW4z+9c8p1QnQ95J8tskQaPlxt86sf0y3Dh28UGFoC7IXM5sIV70BHzg
jDzbqv/y6WFHshAtXS5qvjWd8a816Rj5IFK+UdjUnki6R1vV+/tgVZzI+OfRABjq
V/lfDtuFnjc/H+/r0AYWepeIs80cu1RhJCE/uk90L0u4y3ACc9+otufEsk587R6R
dirJaFTSwJS2Rebrpa0YrRm1JC4hkm9L443UPa9rrwZmL/AQdB8tjzNsk10SPJkY
NYJow0c4SIKHC+VdkqKsPk1IGPPZ53zt70eGQLagg1wCcGUBbUvXjh1ZXwARAQAB
tCFBcnR1c1BCZWhdW92IDxBcnR1ZWthQH1hbmR1eC5ydT6JA1EEEWElADsWlQSn
Yyp+so4AdiQQ/1uSp3a5LPVTRgUCZnU9BgIbAwULCQgHAgI1AgYVCgkICwIEFgID
AQIEBwIXgAAKCRCSp3a5LPVTRpFiD/9WmXVE9a842kHxukXpyJmOUWibfcATOEGl
Ww+m1Mj/ac81H6kd7KubtLGh7VtCoKcWLS0siY680RZ0t3dt/7Tsda0dXSMcZMqe
OFAJkpJa6B1EXEg02+0Lt35g1PNiIRZUoBgbhQXECU+YUu3mYA0LFvJFYkYrr8dg
hQIMHdz+i+ek/Tjx0SPixwqVI9JJYeUosbypRJ69TvcJucONNMKAS5E+U3f1f8cL
jzK2cB0DoBFM46RjI8WTCVEGS4qj/KK6YPu7ndWdXUgoTXQLXx/KydRRNK0Ket0X
```

Рис. 3.5: Копирование ключа

Перейдя в GitHub в браузере, авторизуюсь и добавляю новый GPG ключ под названием Sway, куда вставляю из буфера созданный ключ. (рис. 3.6).

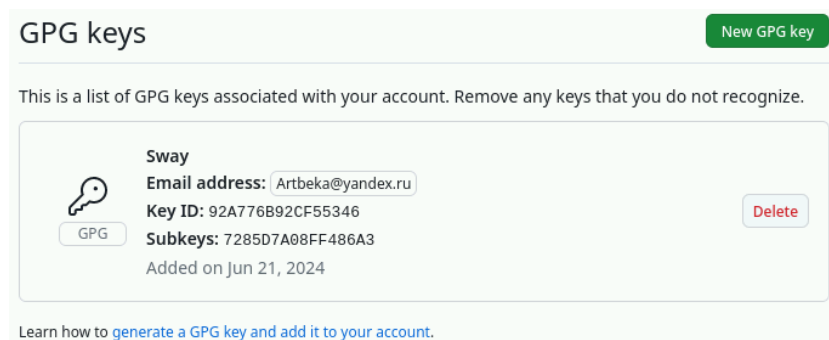


Рис. 3.6: Добавление GPG ключа в GitHub

Далее настраиваю автоматическую подпись коммитов. (рис. 3.7).

```
[atbekauov@atbekauov ~]$ git config --global user.signingkey 92A776B92CF55346
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)

[atbekauov@atbekauov ~]$
```

Рис. 3.7: Автоматические подписи коммитов

Затем провожу авторизацию в github, выбираю авторизоваться через браузер и вхожу там в свой аккаунт github. (рис. 3.8).

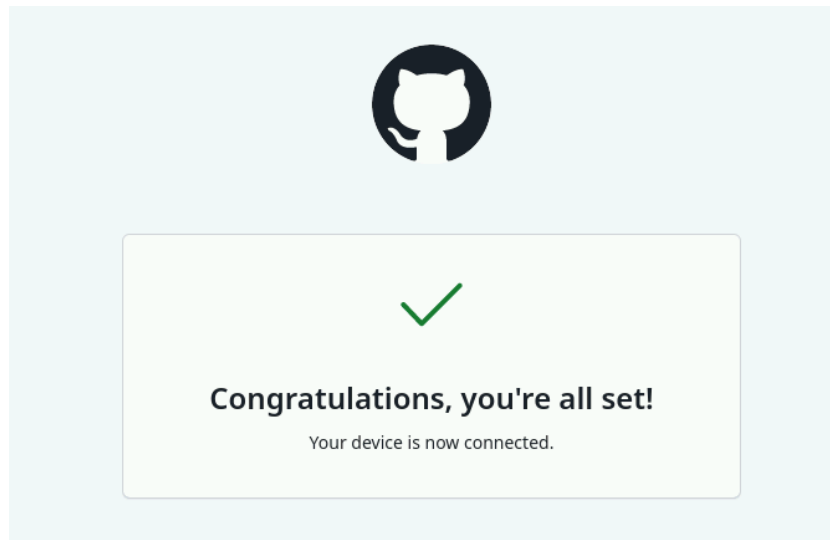


Рис. 3.8: Авторизация устройства в GH

После создаю репозиторий в GH по шаблону приложенному в методичке. (рис. 3.9).

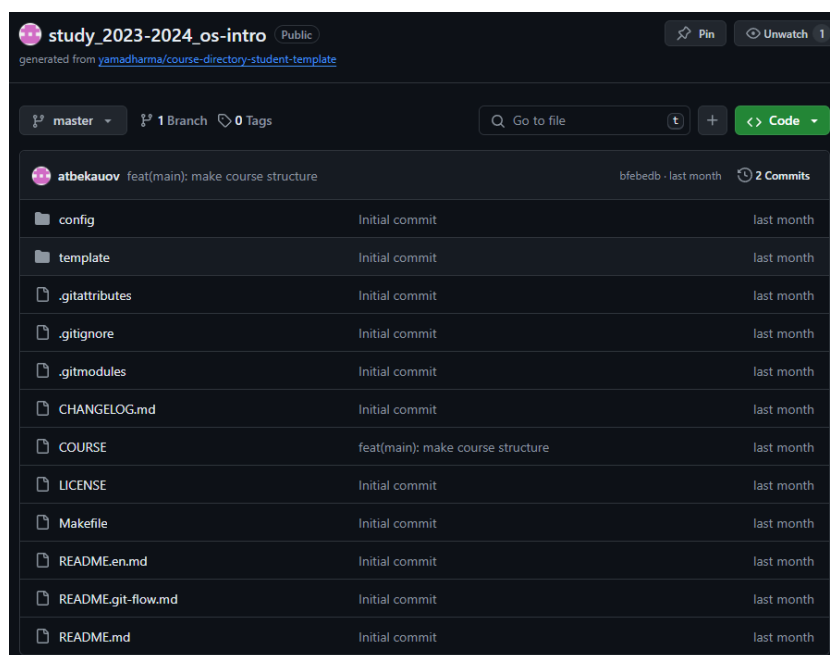


Рис. 3.9: Создание репозитория по шаблону

Далее создаю каталог ~/work/study/2022-2023/“Операционные системы” (рис. 3.10).

```
[atbekauov@atbekauov ~]$
[atbekauov@atbekauov ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
cd ~/work/study/2022-2023/"Операционные системы"
gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
```

Рис. 3.10: Создание каталога для репозитория

Далее я создаю ssh ключи. (рис. 3.11).

```
[atbekauov@atbekauov ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/atbekauov/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/atbekauov/.ssh/id_rsa
Your public key has been saved in /home/atbekauov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:TeeBsSK3RBP5iIK3QrLBIESVyAXdK9a9BAbFmiaTqaI atbekauov@atbekauov
The key's randomart image is:
+---[RSA 4096]-----+
|..=0=. +0.      |
|. + 0 = ... +   |
|0.+.* =.+0+ 0   |
|+*oB00.*.=.o .  |
|0=*..0. S . .   |
|+. . .         |
|o .            |
|E              |
|               |
+----[SHA256]-----+
[atbekauov@atbekauov ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 3.11: Создание ssh ключей

Скопировал ключ в буфер обмена и добавил на GH с названием sway. (рис. 3.12).

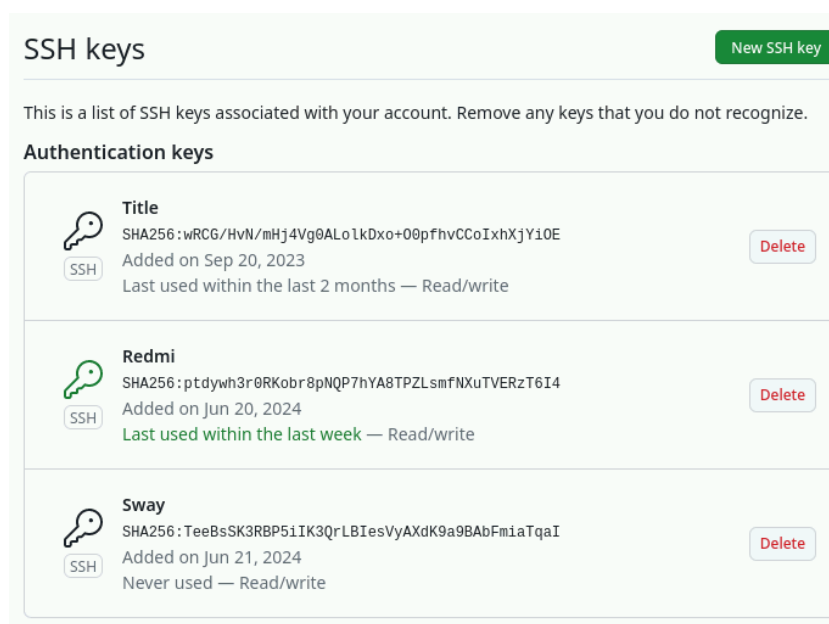


Рис. 3.12: Добавление ssh ключа

После этого клонирую созданный репозиторий в подготовленную ранее папку (локальный репозиторий называю os-intro) (рис. 3.13).

```
atbekauov@atbekauov Операционные системы$ git clone --recursive git@github.com:atbekauov/study_2022-2023-os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.60 KiB | 200.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/atbekauov/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 67 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.39 KiB | 1.24 MiB/c, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/atbekauov/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.88 KiB | 2.35 MiB/c, готово.
Определение изменений: 100% (52/52), готово.
submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
submodule path 'template/report': checked out '7c31ab8e5dfa8c0b2d67caeb8a19ef8028ced88e'
atbekauov@atbekauov Операционные системы$
```

Рис. 3.13: Копирование репозитория курса

Произвожу описанные в методичке действия с файлами репозитория и отправляю изменения на сервер (рис. 3.14).

```

[atbekauov@atbekauov Операционные системы]$ cd os-intro/
[atbekauov@atbekauov os-intro]$ rm package.json
[atbekauov@atbekauov os-intro]$ echo os-intro > COURSE
[atbekauov@atbekauov os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules

[atbekauov@atbekauov os-intro]$
[atbekauov@atbekauov os-intro]$ git add .
[atbekauov@atbekauov os-intro]$ git commit -am 'feat(main): make course structure'
[master 3932bc9] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
[atbekauov@atbekauov os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 947 байтов | 947.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:atbekauov/study_2022-2023_os-intro.git
   329e52f..3932bc9  master -> master
[atbekauov@atbekauov os-intro]$

```

Рис. 3.14: Отправка изменений на сервер

## 4 Выводы

В ходе данной лабораторной работы я изучил идеологии и применения средств контроля версий, освоил умения по работе с git.

## 5 Ответы на контрольные вопросы

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого

репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. 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 в текущей директории: `git status`

Просмотр текущих изменений: `git diff`

Сохранение текущих 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 (при этом файл и/или каталог 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 на текущей: `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 с текущим 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. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.