

Отчёт по лабораторной работе №2

Первоначальна настройка git

Ганина Таисия Сергеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Системы контроля версий. Общие понятия	7
4	Выполнение лабораторной работы	12
5	Выводы	27
	Список литературы	28

Список иллюстраций

4.1	Базовая настройка git	12
4.2	Создание ключа по алгоритму rsa с ключём размером 4096 бит . .	12
4.3	Создание ключа по алгоритму ed25519	13
4.4	Команда начала генерации ключа	13
4.5	Настройка параметров	14
4.6	Настройка параметров	14
4.7	Создание пароля	15
4.8	Генерация	15
4.9	Вывод списка ключей	16
4.10	Загрузка ключа	16
4.11	Загрузка ключа	16
4.12	Итог	17
4.13	Настройка автоматических подписей коммитов git	17
4.14	Авторизация	17
4.15	Подключение к браузеру	18
4.16	Итог	18
4.17	Создание репозитория на основе шаблона	19
4.18	Репозиторий	19
4.19	Авторизация	20
4.20	Подключение к браузеру	20
4.21	Итог	21
4.22	Переход в каталог, удаление лишних файлов, создание каталогов .	21
4.23	Отправка файлов на сервер	22
4.24	Отправка файлов на сервер	22
4.25	Отправка файлов на сервер	23

Список таблиц

3.1	Описание основных команд системы git	9
-----	--	---

1 Цель работы

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

2 Задание

1. Базовая настройка git.
2. Создание SSH ключа.
3. Создание PGP ключа.
4. Добавление PGP ключа в GitHub.
5. Настройка автоматических подписей коммитов git.
6. Настройка gh.
7. Шаблон для рабочего пространства.
8. Создание локального репозитория курса на основе шаблона.
9. Настройка каталога курса.
10. Контрольные вопросы.

3 Теоретическое введение

3.1 Системы контроля версий. Общие понятия

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 зависимости 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.

Примеры использования git

- Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
- Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

Например, в табл. 3.1 приведено описание основных команд системы git.

Таблица 3.1: Описание основных команд системы git

Команда	Действие
git init	Создание основного дерева репозитория
git pull	Получение обновлений (изменений) текущего дерева из центрального репозитория
git push	Отправка всех произведённых изменений локального дерева в центральный репозиторий
git status	Просмотр списка изменённых файлов в текущей директории
git diff	Просмотр текущих изменений
git add .	Сохранение текущих изменений
/ git add	
<имя файла>	
/ git rm	
<имя файла>	

Команды	
Команда	Действие
git commit / git commit -am "описание коммита"	Сохранение добавленных изменений
git checkout -b имя_ветки	Создание новой ветки, базирующейся на текущей
git checkout имя_ветки	Переключение на некоторую ветку
git push origin имя_ветки	Отправка изменений конкретной ветки в центральный репозиторий
git merge --no-ff имя_ветки	Слияние ветки с текущим деревом
git branch -d имя_ветки	Удаление локальной уже слитой с основным деревом ветки

Команда	
Команда	Действие
git branch -D имя_ветки	Принудительное удаление локальной ветки
git push origin :имя_ветки	Удаление ветки с центрального репозитория

4 Выполнение лабораторной работы

1. Базовая настройка git (рис. 4.1).

```
[tsganina@tsganina Операционные системы]$ cd ~  
[tsganina@tsganina ~]$ git config --global user.name "tsganina"  
[tsganina@tsganina ~]$ git config --global user.email "tai.sergeevna@yandex.ru"  
[tsganina@tsganina ~]$ git config --global core.quotepath false  
[tsganina@tsganina ~]$ git config --global init.defaultBranch master  
[tsganina@tsganina ~]$ git config --global core.autocrlf input  
[tsganina@tsganina ~]$ git config --global core.safecrlf warn  
[tsganina@tsganina ~]$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.
```

Рис. 4.1: Базовая настройка git

2. Создание SSH ключа (рис. 4.2, 4.3).

```
[tsganina@tsganina Операционные системы]$ cd ~  
[tsganina@tsganina ~]$ git config --global user.name "tsganina"  
[tsganina@tsganina ~]$ git config --global user.email "tai.sergeevna@yandex.ru"  
[tsganina@tsganina ~]$ git config --global core.quotepath false  
[tsganina@tsganina ~]$ git config --global init.defaultBranch master  
[tsganina@tsganina ~]$ git config --global core.autocrlf input  
[tsganina@tsganina ~]$ git config --global core.safecrlf warn  
[tsganina@tsganina ~]$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.
```

Рис. 4.2: Создание ключа по алгоритму rsa с ключём размером 4096 бит

```

[tsganina@tsganina ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/tsganina/.ssh/id_ed25519):
/home/tsganina/.ssh/id_ed25519 already exists.
Overwrite (y/n)?
[tsganina@tsganina ~]$ y
bash: y: команда не найдена...
[tsganina@tsganina ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/tsganina/.ssh/id_ed25519):
/home/tsganina/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y

```

Рис. 4.3: Создание ключа по алгоритму ed25519

3. Создание PGP ключа (рис. 4.4, 4.5, 4.6, 4.7, 4.8, 4.9).

```

[tsganina@tsganina ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.8; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/tsganina/.gnupg'
gpg: создан щит с ключами '/home/tsganina/.gnupg/pubring.kbx'
Выберите тип ключа:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (только для подписи)
(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072)

```

Рис. 4.4: Команда начала генерации ключа

```
Терминал - tsganina@tsganina:~
Файл  Правка  Вид  Терминал  Вкладки  Справка

(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (только для подписи)
(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
    0 = не ограничен
    <n> = срок действия ключа - n дней
    <n>w = срок действия ключа - n недель
    <n>m = срок действия ключа - n месяцев
    <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: 
```

Рис. 4.5: Настройка параметров

```
Терминал - tsganina@tsganina:~
Файл  Правка  Вид  Терминал  Вкладки  Справка

(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
    0 = не ограничен
    <n> = срок действия ключа - n дней
    <n>w = срок действия ключа - n недель
    <n>m = срок действия ключа - n месяцев
    <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: tsganina
Адрес электронной почты: tai.sergeevna@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "tsganina <tai.sergeevna@yandex.ru>"
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
```

Рис. 4.6: Настройка параметров

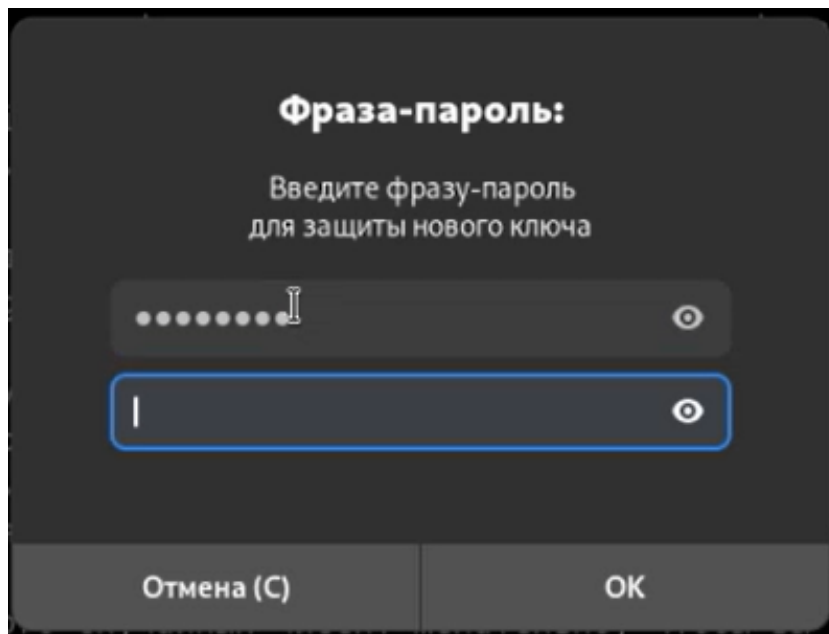


Рис. 4.7: Создание пароля

```

Терминал - tsganina@tsganina:~
Файл Правка Вид Терминал Вкладки Справка
Вы выбрали следующий идентификатор пользователя:
"tsganina <tai.sergeevna@yandex.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/tsganina/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/tsganina/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/tsganina/.gnupg/openpgp-revocs.d/0DA8CD5
657438A08749880AD56E22FEB63FFAA68.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-02-15 [SC]
       0DA8CD5657438A08749880AD56E22FEB63FFAA68
uid                               tsganina <tai.sergeevna@yandex.ru>
sub   rsa4096 2023-02-15 [E]

[tsganina@tsganina ~]$

```

Рис. 4.8: Генерация

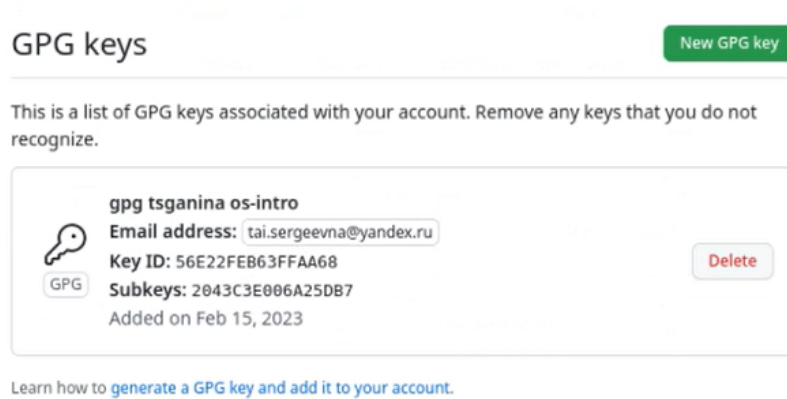


Рис. 4.12: Итог

5. Настройка автоматических подписей коммитов git (рис. 4.13).

```
[tsganina@tsganina ~]$ git config --global user.signingkey 56E22FEB63FFAA68
[tsganina@tsganina ~]$ git config --global commit.gpgsign true
[tsganina@tsganina ~]$ git config --global gpg.program $(which gpg2)
[tsganina@tsganina ~]$
```

Рис. 4.13: Настройка автоматических подписей коммитов git

6. Настройка gh (рис. 4.14, 4.15, 4.16).

```
[tsganina@tsganina ~]$ gh auth login
? You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/tsganina/.ssh/id_rsa.
pub
? Title for your SSH key: tsganina 19 04 new key
? How would you like to authenticate GitHub CLI? Login with a web browser
! First copy your one-time code: 3580-AFB2
Press Enter to open github.com in your browser...
```

Рис. 4.14: Авторизация

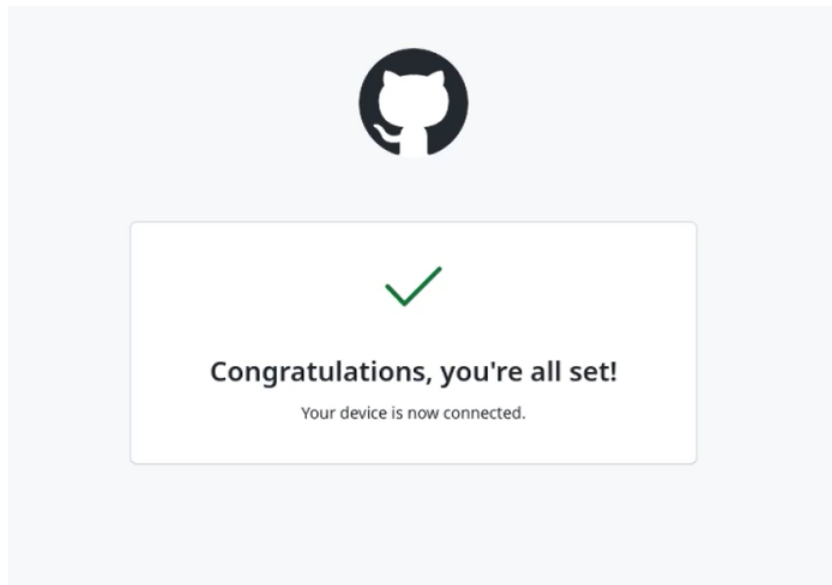


Рис. 4.15: Подключение к браузеру

```
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/tsganina/.ssh/id_rsa.pub
✓ Logged in as tsganina
[tsganina@tsganina ~]$
```

Рис. 4.16: Итог

7. Шаблон для рабочего пространства (рис. 4.17, 4.18).

Create a new repository from course-directory-student-template

The new repository will start with the same files and folders as [yamadharma/course-directory-student-template](#).

Owner * tsganina / Repository name * study_2022-2023_os-intro ✓

Great repository names are short and memorable. Need inspiration? How about [stunning-guide](#)?

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.

☐ **Include all branches**
Copy all branches from yamadharma/course-directory-student-template and not just master.

i You are creating a public repository in your personal account.

Creating repository

Рис. 4.17: Создание репозитория на основе шаблона

master 1 branch 0 tags Go to file Add file Code

tsganina	Initial commit	c788d86 now 1 commit
config	Initial commit	now
template	Initial commit	now
.gitattributes	Initial commit	now
.gitignore	Initial commit	now
.gitmodules	Initial commit	now
CHANGELOG.md	Initial commit	now
COURSE	Initial commit	now
LICENSE	Initial commit	now
Makefile	Initial commit	now
README.en.md	Initial commit	now
README.git-flow.md	Initial commit	now
README.md	Initial commit	now
package.json	Initial commit	now

Рис. 4.18: Репозиторий

- Создание локального репозитория курса на основе шаблона (рис. 4.19, 4.20, 4.21).

```
[tsganina@tsganina ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"  
[tsganina@tsganina ~]$ cd ~/work/study/2022-2023/"Операционные системы"  
[tsganina@tsganina Операционные системы]$
```

Рис. 4.19: Авторизация

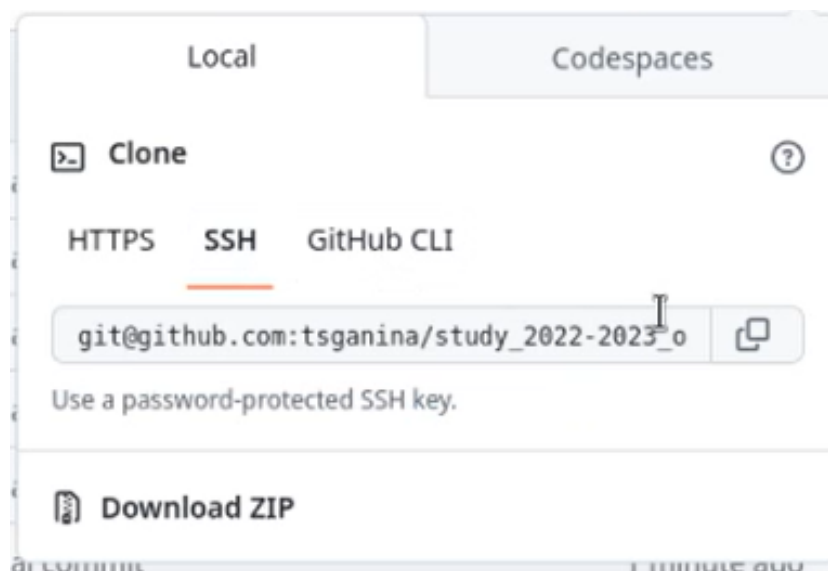
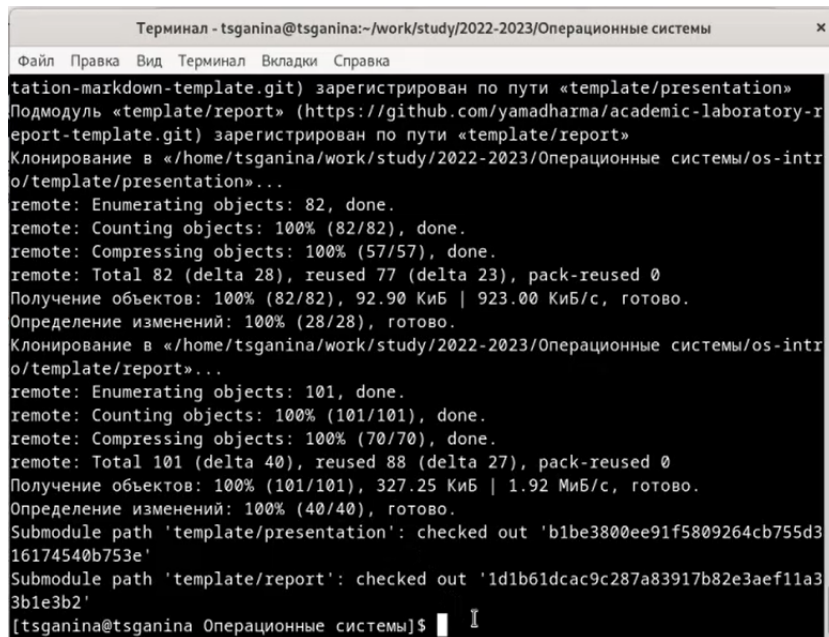


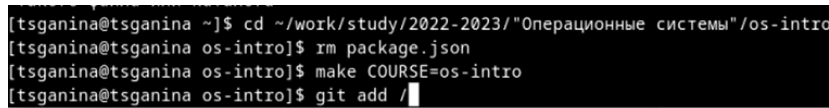
Рис. 4.20: Подключение к браузеру



```
Терминал - tsganina@tsganina:~/work/study/2022-2023/Операционные системы
Файл  Правка  Вид  Терминал  Вкладки  Справка
tation-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/tsganina/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 923.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/tsganina/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 1.92 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
[tsganina@tsganina Операционные системы]$
```

Рис. 4.21: Итог

9. Настройка каталога курса (рис. 4.22, 4.23, 4.24, 4.25).



```
[tsganina@tsganina ~]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[tsganina@tsganina os-intro]$ rm package.json
[tsganina@tsganina os-intro]$ make COURSE=os-intro
[tsganina@tsganina os-intro]$ git add /
```

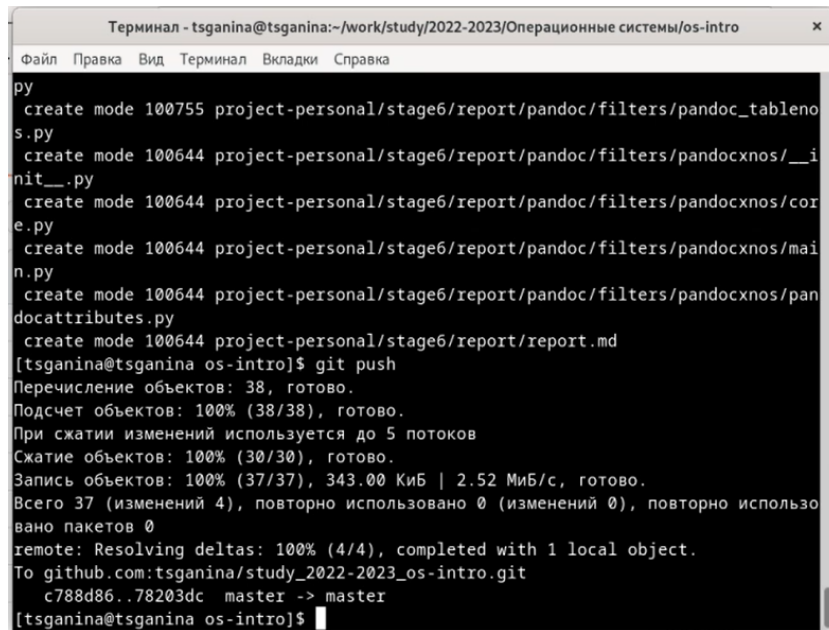
Рис. 4.22: Переход в каталог, удаление лишних файлов, создание каталогов

```
Терминал - tsganina@tsganina:~/work/study/2022-2023/Операционные системы/os-intro
Файл Правка Вид Терминал Вкладки Справка
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
[tsganina@tsganina os-intro]$
```

Рис. 4.23: Отправка файлов на сервер

tsganina feat(main): make course structure ...			now 2
config	Initial commit		6 minutes ago
labs	feat(main): make course structure		now
presentation	feat(main): make course structure		now
project-personal	feat(main): make course structure		now

Рис. 4.24: Отправка файлов на сервер



```
Терминал - tsganina@tsganina:~/work/study/2022-2023/Операционные системы/os-intro
Файл Правка Вид Терминал Вкладки Справка
ру
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
[tsganina@tsganina os-intro]$ git push
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 5 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 343.00 КиБ | 2.52 МБ/с, готово.
Всего 37 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:tsganina/study_2022-2023_os-intro.git
c788d86..78203dc master -> master
[tsganina@tsganina os-intro]$
```

Рис. 4.25: Отправка файлов на сервер

10. Контрольные вопросы.

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

Система контроля версий - программное обеспечение для облегчения работы с изменяющейся информацией.

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.

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

- Хранилище - репозиторий, хранилище версий, в нем хранятся все документы вместе с их историей, и другой служебной информацией.
 - Commit - отслеживание изменений, сохранение разницы в рабочих изменениях.
 - История - сохранение всех изменений в проекте и при необходимости возможность обратиться к старым данным.
 - Рабочая копия - копия проекта, связанная с репозиторием, текущее состояние файлов проекта, основанное на их последней версии из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
- Централизованные VCS - одно основное хранилище всего проекта, каждый пользователь копирует себе необходимые файлы из репозитория и изменяет их, а потом может добавить обратно. Примеры: - Subversion; - CVS; - TFS; - VAULT; - AccuRev;
 - Децентрализованные VCS - У каждого пользователя свой вариант репозитория, есть возможность добавлять и забирать версии из любого репозитория. Например: - Git; - Mercurial; - Bazaar;

Обычно используются централизованные системы контроля версий, с одним общим репозиторием.

4. Опишите действия с VCS при единоличной работе с хранилищем. Сначала нужно создать удалённый репозиторий, после - подключить его. Затем по мере того, как проект будет выполняться, нужно отправлять данные на сервер.
5. Опишите порядок работы с общим хранилищем VCS. В начале при помощи последовательности команд пользователь получает нужную ему версию данных. Потом он работает с ними, вносит некоторые изменения, и уже

после этого он может разместить новую версию в хранилище. При этом предыдущие версии не удаляются из хранилища, что является очень удобным - к ним можно вернуться в любой момент.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Хранить информацию обо всех изменениях, производимых в проекте.
- Обеспечить командную работу.

7. Назовите и дайте краткую характеристику командам git.

`git init` - Создание основного дерева репозитория

`git pull` - Получение обновлений (изменений) текущего дерева из центрального репозитория

`git push` - Отправка всех произведённых изменений локального дерева в центральный репозиторий

`git status` - Просмотр списка изменённых файлов в текущей директории изменений

`git add .` / `git add <имя файла>` / `git rm <имя файла>` - Сохранение текущих изменений

`git commit` / `git commit -am "описание коммита"` - Сохранение добавленных изменений

`git checkout -b имя_ветки` - Создание новой ветки, базирующейся на текущей

`git checkout имя_ветки` - Переключение на некоторую ветку

`git push origin имя_ветки` - Отправка изменений конкретной ветки в центральный репозиторий

`git merge --no-ff имя_ветки` - Слияние ветки с текущим деревом

`git branch -d имя_ветки` - Удаление локальной уже слитой с основным деревом ветки

`git branch -D имя_ветки` - Принудительное удаление локальной ветки

`git push origin :имя_ветки` - Удаление ветки с центрального репозитория.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями. `git push -all` (`push origin master/любой branch`)
9. Что такое и зачем могут быть нужны ветви (branches)? Ветвь - один из параллельных участков истории проекта в одном хранилище. Все ветви исходят из одной версии - точки ветвления. Обычно ветви делятся на `master` и `trunk`. Между ветками возможно и слияние. Ветки нужны для разработки новых функций.
10. Как и зачем можно игнорировать некоторые файлы при `commit`? Игнорировать некоторые файлы можно прописав шаблон `.gitignore` специально для игнорируемых файлов. Зачем это нужно? Чтобы в репозиторий не попали “лишние” файлы, которые неминуемо будут возникать при работе над проектом. Это могут быть временные файлы, объектные файлы.

5 Выводы

- Я изучила идеологию и применение средств контроля версий.
- Освоила умения по работе с git.

Список литературы

1. Руководство к выполнению лабораторной работы №2
2. Solving “Fatal: Not A Git Repository” (Or Any Of The Parent Directories) Error