

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Максим Александрович Мишонков

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	17

Список иллюстраций

4.1	Создание каталога	7
4.2	Каталог “Операционные системы”	7
4.3	Создание репозитория	7
4.4	Создание репозитория	8
4.5	Создание репозитория	8
4.6	Создание репозитория	9
4.7	Создание репозитория	10
4.8	Создание репозитория	10
4.9	Созданный репозиторий	10
4.10	Клонирование созданного репозитория	11
4.11	Удаление лишних файлов и создание необходимых каталогов . .	11
4.12	Созданные каталоги	11
4.13	Отправка файлов на сервер	11
4.14	Отправка файлов на сервер	11
4.15	Отправка файлов на сервер	12
4.16	Отправка файлов на сервер	12
4.17	Отправленные на сервер файлы	12

1 Цель работы

Целью данной лабораторной работы является изучение идеологии и применения средств системы контроля версий Git, а также приобретение практических навыков работы с ней.

2 Задание

Научиться работать с системой контроля версий Git.

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.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом.

4 Выполнение лабораторной работы

1. Создал каталог “Операционные системы”. (рис. [4.1], [4.2])

```
mamishonkov@dk5n51 ~$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
```

Рис. 4.1: Создание каталога

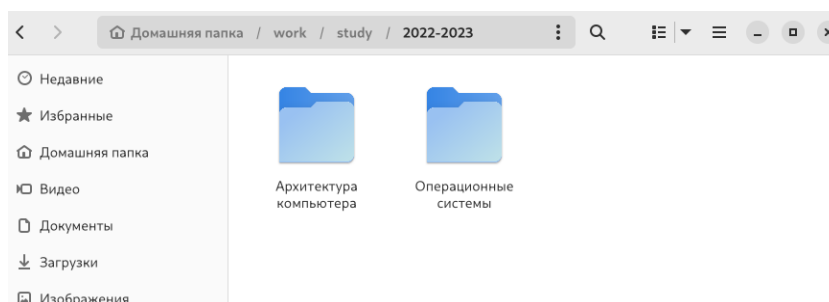


Рис. 4.2: Каталог “Операционные системы”

2. Перешёл в каталог “Операционные системы” и начал процесс создания репозитория на GitHub. (рис. [4.3], [4.4], [4.5], [4.6], [4.7], [4.8], [4.9])

```
mamishonkov@dk5n51 ~$ cd ~/work/study/2022-2023/"Операционные системы"
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
To get started with GitHub CLI, please run: gh auth login
Alternatively, populate the GH_TOKEN environment variable with a GitHub API authentication token.
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
https://github.com/mishonkovm
! Sorry, your reply was invalid: "https://github.com/mishonkovm" is not a valid answer, please try again.
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: E944-5B33
Press Enter to open github.com in your browser...
█
```

Рис. 4.3: Создание репозитория

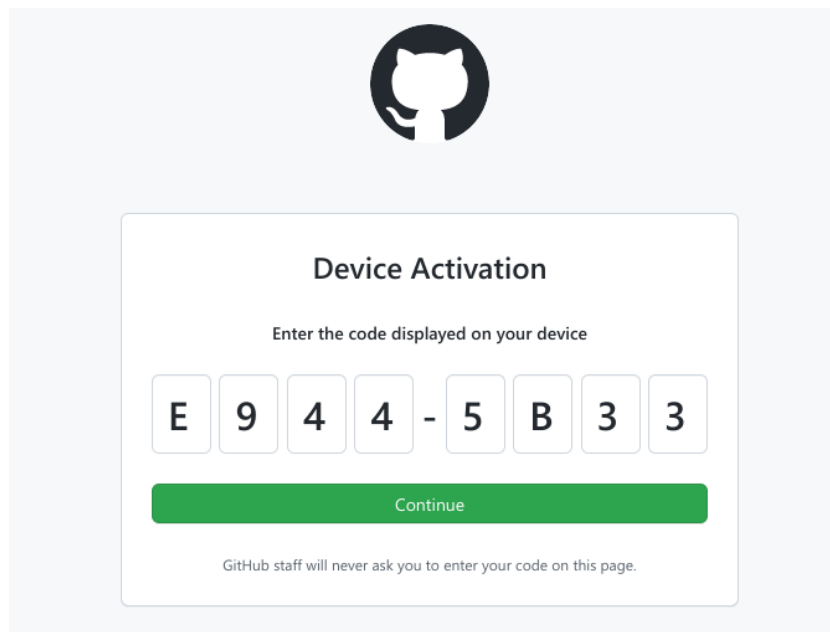


Рис. 4.4: Создание репозитория

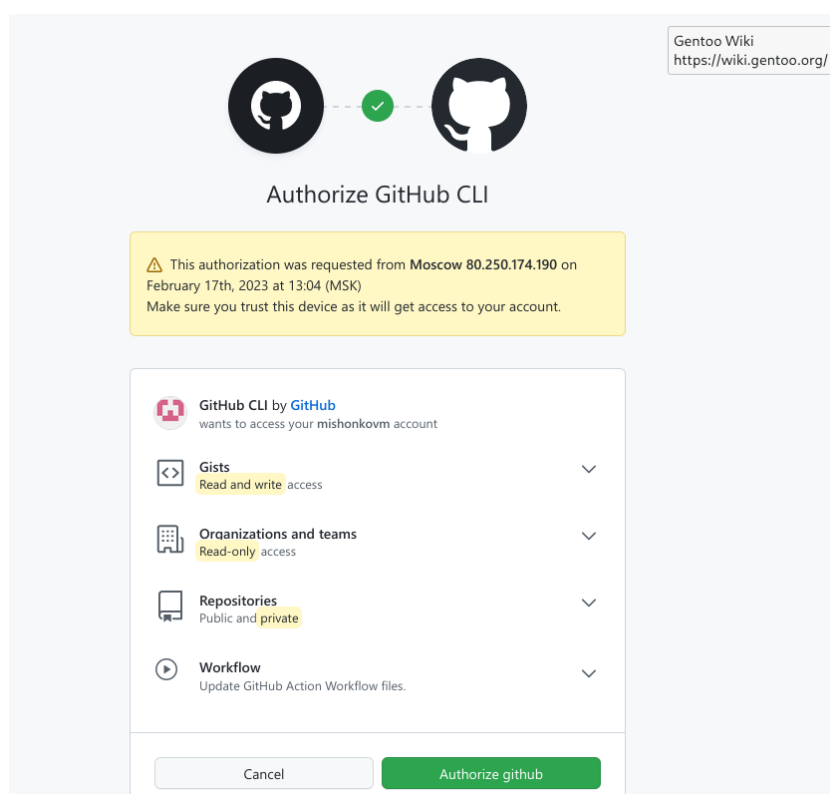


Рис. 4.5: Создание репозитория



Confirm access

Password

[Forgot password?](#)

Confirm

Tip: You are entering [sudo mode](#). After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

Рис. 4.6: Создание репозитория

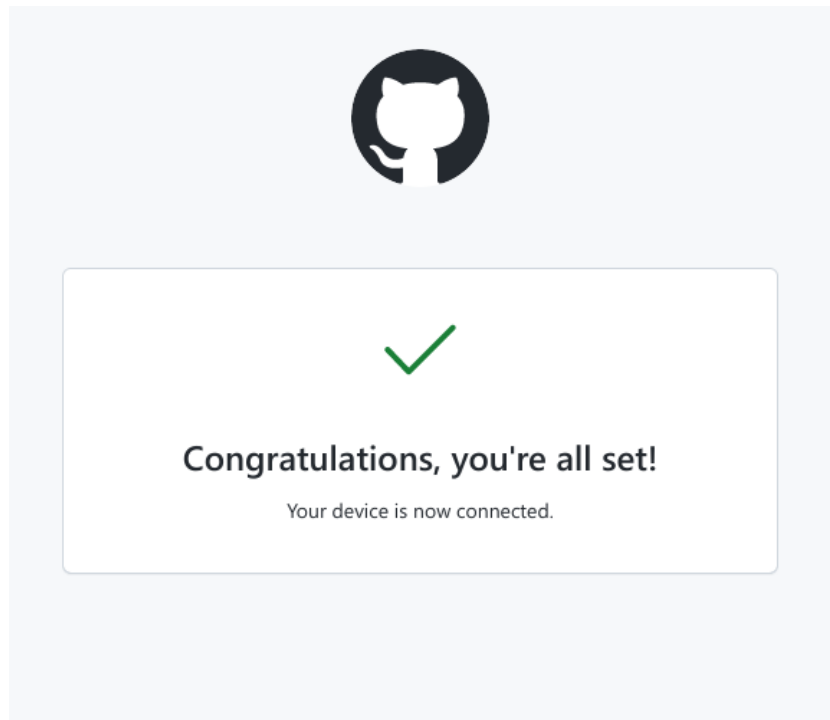


Рис. 4.7: Создание репозитория

```
✓ Created repository mishonkovm/study_2022-2023_os-intro on GitHub  
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы $
```

Рис. 4.8: Создание репозитория

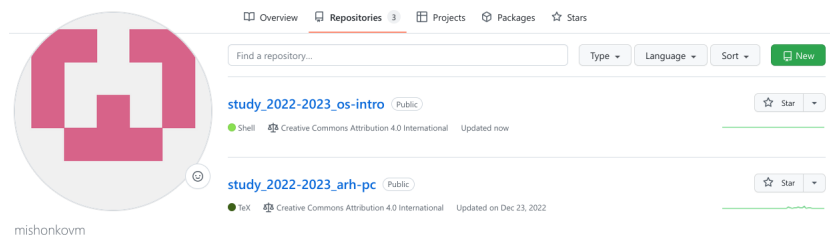


Рис. 4.9: Созданный репозиторий

3. Клонировал созданный репозиторий. (рис. [4.10])

```

mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:mamishonkov/study_2022-2023-os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 27 (delta 1), reused 11 (delta 9), pack-reused 0
Получение объектов: 100% (27/27), 15.93 KiB | 16.93 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подсудить «template/presentation» (https://github.com/yamchurba/academic-presentation-mardown-template.git) зарегистрирован по пути «template/presentation»
Подсудить «template/report» (https://github.com/yamchurba/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/afs/rl.scl.psu.edu/pub/hosts/n/a/mamishonkov/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% (27/27), done.
remote: Total 82 (delta 28), reused 72 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 KiB | 980.89 KiB/c, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/afs/rl.scl.psu.edu/pub/hosts/n/a/mamishonkov/work/study/2022-2023/Операционные системы/os-intro/template/report...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 191 (delta 48), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (191/191), 322.25 KiB | 2.37 MiB/c, готово.
Определение изменений: 100% (49/49), готово.
Submodule path 'template/presentation': checked out 'b1be389bee91f3809264cb755d31517454eb753e'
Submodule path 'template/report': checked out 'd1b51dcac5c287a8337b8263ccf1a33b1c3b2'
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы $

```

Рис. 4.10: Клонирование созданного репозитория

4. Удалил лишние файлы и создал необходимые каталоги. (рис. [4.11], [4.12])

```

mamishonkov@dk5n51 ~ $ cd ~/work/study/2022-2023/Операционные системы/os-intro
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ rm package.json
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro > COURSE
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ make

```

Рис. 4.11: Удаление лишних файлов и создание необходимых каталогов

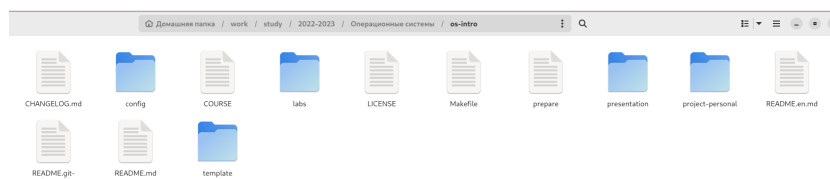


Рис. 4.12: Созданные каталоги

5. Отправил файлы на сервер. (рис. [4.13], [4.14], [4.15], [4.16], [4.17])

```

mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ git add .

```

Рис. 4.13: Отправка файлов на сервер

```

mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'

```

Рис. 4.14: Отправка файлов на сервер

```
[master 87d1a16] feat(main): make course structure
361 files changed, 100327 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab01/report/report.md
```

Рис. 4.15: Отправка файлов на сервер

```
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ git push
Перечисление объектов: 40%, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.39 КиБ | 3.32 МБ/с, готово.
Всего 38 (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:mishonkovm/study_2022-2023_os-intro.git
66e26d7..87d1a16 master -> master
mamishonkov@dk5n51 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'
```

Рис. 4.16: Отправка файлов на сервер

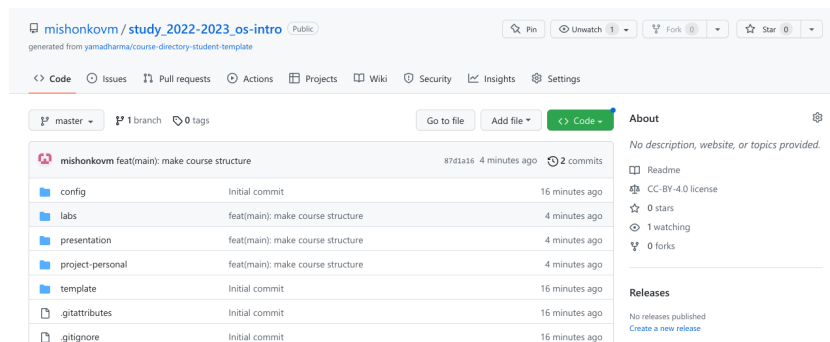


Рис. 4.17: Отправленные на сервер файлы

Ответы на контрольные вопросы

1. Система управления версиями (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного

и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

2. 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.
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 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 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. Основные команды git: 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 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 вет-
 ки, 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 репо-
 зиторий: `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 цен-
 трального 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. Проблемы, которые решают ветки `git`:

- 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, ко-
 торые не требуется добавлять в последствии в репозиторий. Например, вре-
 менные файлы, создаваемые редакторами, или объектные файлы, созда-
 ваемые компиляторами. Можно прописать шаблоны игнорируемых при
 добавлении в репозиторий типов файлов в файл `.gitignore` с помощью ер-
 висов. Для этого сначала нужно получить списки 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`

5 Выводы

В ходе выполнения данной лабораторной работы я создал каталог “Операционные системы” и новый репозиторий на сервере GitHub на основе шаблона, внёс в него изменения. Также я создал рабочее пространство для выполнения следующих лабораторных работ.