РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Студент: Лисенков Е.Р

Группа: НКАбд-03-23

МОСКВА

2023 г.

Содержание

Список иллюстраций

1 Цель работы

Изучить идеологию и применение средств контроля версиями, а также получить практические навыки по работе с системой git.

2 Задания

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблоа.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево п роекта 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, произ ведённые разными участниками проекта, производить откат к любой более ранн ей версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Вы полнение большинства функций по управлению версиями осуществляется специа льным сервером. Участник проекта (пользователь) перед началом работы посред ством определённых команд получает нужную ему версию файлов. После внесе ния изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно в ернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохра нять только изменения между последовательными версиями, что позволяет умен ьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и раз решения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными у частниками (автоматически или вручную), вручную выбрать нужную версию, от менить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволя ет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким об разом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, боле
е гибкие функциональные возможности. Например, они могут поддерживать раб
оту с

несколькими версиями одного файла, сохраняя общую историю изменений
до точки ветвления версий и собственные истории изменений каждой ветви. К
роме того, обычно доступна информация о том, кто из участников, когда и как
ие изменения вносил. Обычно такого рода информация хранится в журнале изм
енений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий ц
ентральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди рас
пределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются
они в основном синтаксисом используемых в работе команд.

Система контроля версий Git представляет собой набор программ коман
дной строки. Доступ к ним можно получить из терминала посредством ввода ко
манды git с различными опциями.

Благодаря тому, что Git является распределённой системой контроля версий, рез
ервную копию локального хранилища можно сделать простым копированием ил
и архивацией. **Система контроля версий Git** представляет собой набор програм
м командной строки. Доступ к ним можно получить из терминала посредством
ввода команды git с различными опциями. Благодаря тому, что Git является рас
пределённой системой контроля версий, резервную копию локального хранилища
можно сделать простым копированием или архивацией.

4 Выполнение лабораторной работы

4.1 Настройка GitHub

Первым делом создам учётную запись на сайте GitHub и всё заполню (рис. 4.1).

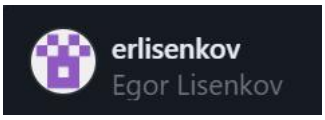


Рис. 4.1 Аккаунт GitHub

4.2 Базовая настройка GitHub

После открытия виртуальной машины, начинаю настройку с помощью: git config –global user.name “” И git config –global user.email “work@mail” куда я ввожу свою почту (рис. 4.2).

```
[erlisenkov@fedora ~]$ git config --global user.name "Egor Lisenkov"
[erlisenkov@fedora ~]$ git config --global user.email "1132232881@pfur.ru"
```

Рис. 4.2 Конфигурация Git

Далее настраиваю utf-8 в выводе сообщений git для отображения символов (рис. 4.3).

```
[erlisenkov@fedora ~]$ git config --global core.quotepath false
```

Рис. 4.3 Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 4.4).

```
[erlisenkov@fedora ~]$ git config --init.defaultBranch master
```

Рис. 4.4 Создание имени для начальной ветки

Задам параметр autocrlf со значением input, для того чтобы конвертировать CRLF в LF только при коммитах (рис. 4.5).

```
[erlisenkov@fedora ~]$ git config --global core.autocrlf input
```

Рис. 4.5 Параметр autocrlf

Задам параметр safeclf со значением warm, чтобы Git мог проверять преобразования на обратимость (рис. 4.6).

```
[erlisenkov@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 4.6 Параметр safecrlf

4.3 Создание SSH-ключа

Чтобы идентифицировать пользователя на сервере репозиториев нужно сделать пару ключей с помощью команд: ssh-keygen -C “Имя Фамилия, work@email” (рис. 4.7).

```
[erlisenkov@fedora ~]$ ssh-keygen -C"Egor Lisenkov, 1132232881@pfur.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/erlisenkov/.ssh/id_rsa):
Created directory '/home/erlisenkov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/erlisenkov/.ssh/id_rsa
Your public key has been saved in /home/erlisenkov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:k5taRjIh2hoiKzziakSIXd13dUOw2U/KMg0fDJv5Zgc Egor Lisenkov, 1132232881@pfu
r.ru
The key's randomart image is:
+---[RSA 3072]----+
|      . .    .o+o.|
|     . . . . .*+..|
|o. .. .  . .=oE..|
|o..o . . .   * =.|
|+ o . o S   o O o|
|o+ o   + +   = . |
|=o.     =        |
|+..    +         |
|+.    .          |
+----[SHA256]-----+
[erlisenkov@fedora ~]$ S
```

Рис. 4.7 Генерация SSH-ключа

Далее установим утилиту Xclip, которая позволяет нам скопировать любой текст через терминал. На Linux мы должны скачать её, поэтому проделаем ко манды, которые вы увидите у меня на скриншотах (рис. 4.8 - рис. 4.11).

```
[erlisenkov@fedora ~]$ Scat ~/.ssh/id_rsa.pub | xclip -sel clip
bash: Scat: команда не найдена...
bash: xclip: команда не найдена...
Аналогичная команда: 'cat'
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y


 * Ожидание в очереди...
 * Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
 xclip-0.13-19.git11cba61.fc38.x86_64   Command line clipboard grabber
Продолжить с этими изменениями? [N/y] y


 * Ожидание в очереди...
 * Ожидание аутентификации...
 * Ожидание в очереди...
 * Загрузка пакетов...
 * Запрос данных...
 * Проверка изменений...
 * Установка пакетов...
```

Рис. 4.8 Установка Xclip

```
[erlisenkov@fedora ~]$ sudo dnf makecache --refresh
Copr repo for PyCharm owned by phracek 2.5 kB/s | 2.1 kB     00:00
Fedora 38 - x86_64                      48 kB/s |  19 kB     00:00
Fedora 38 openh264 (From Cisco) - x86_ 1.5 kB/s | 989  B     00:00
Fedora Modular 38 - x86_64              44 kB/s |  19 kB     00:00
Fedora 38 - x86_64 - Updates            20 kB/s |  14 kB     00:00
Fedora 38 - x86_64 - Updates           1.4 MB/s | 3.0 MB     00:02
Fedora Modular 38 - x86_64 - Updates    21 kB/s |  17 kB     00:00
google-chrome                          4.9 kB/s | 1.3 kB     00:00
RPM Fusion for Fedora 38 - Nonfree - N 6.9 kB/s | 6.8 kB     00:00
RPM Fusion for Fedora 38 - Nonfree - S  16 kB/s | 6.5 kB     00:00
Создан кэш метаданных.
[erlisenkov@fedora ~]$
```

Рис. 4.9 Установка Xclip

```
[erlisenkov@fedora ~]$ sudo dnf -y install xclip
Последняя проверка окончания срока действия метаданных: 0:00:47 назад,
Ср 20 сен 2023 10:20:16.
Пакет xclip-0.13-19.git11cba61.fc38.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[erlisenkov@fedora ~]$
```

Рис. 4.10 Установка Xclip

Делаю запрос на ключ SSH, который потребуется нам для GitHub (рис. 4.11).

```
[erlisenkov@fedora ~]$ cat ~/.ssh/id_rsa.pub
```

Рис. 4.11 Запрос ключа SSH

Перехожу на сайт GitHub и прикрепляю ключ для соединения с GitHub (рис. 4.12).

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2022-2023/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. 4.12).

```
[erlisenkov@fedora ~]$ mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
[erlisenkov@fedora ~]$ ls
 work    Документы   Изображения   Общедоступные   Шаблоны
 Видео   Загрузки    Музыка        'Рабочий стол'
[erlisenkov@fedora ~]$ S
```

Рис. 4.12 Создание рабочего пространства

4.5 Создание репозитория курса на основе шаблона

Выполню переход по ссылке https://github.com/yamadharma/course-directory-student-template и выбираю «Use this tamplate», чтобы использовать шаблон для своего репозитория (рис. 4.13).

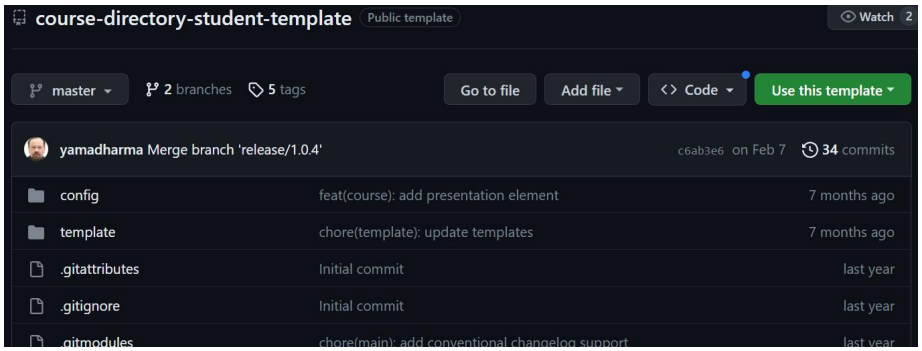


Рис. 4.13 Страница шаблона

В открывшемся окне нажимаю на кнопку «Create repository» (рис. 4.14).

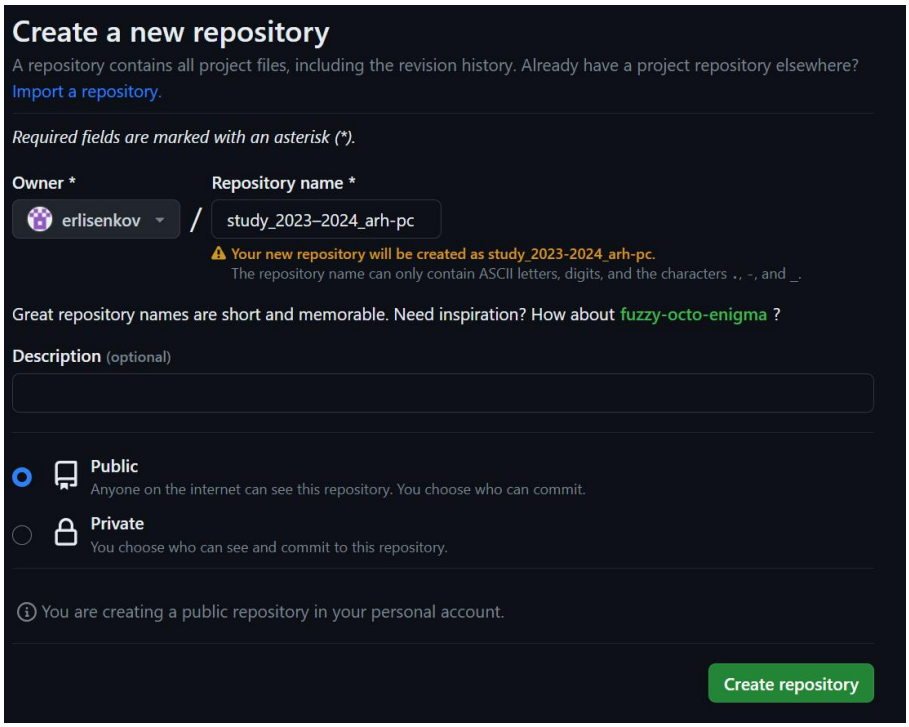


Рис. 4.14 Создание репозитория

Созданный репозиторий (рис. 4.15).

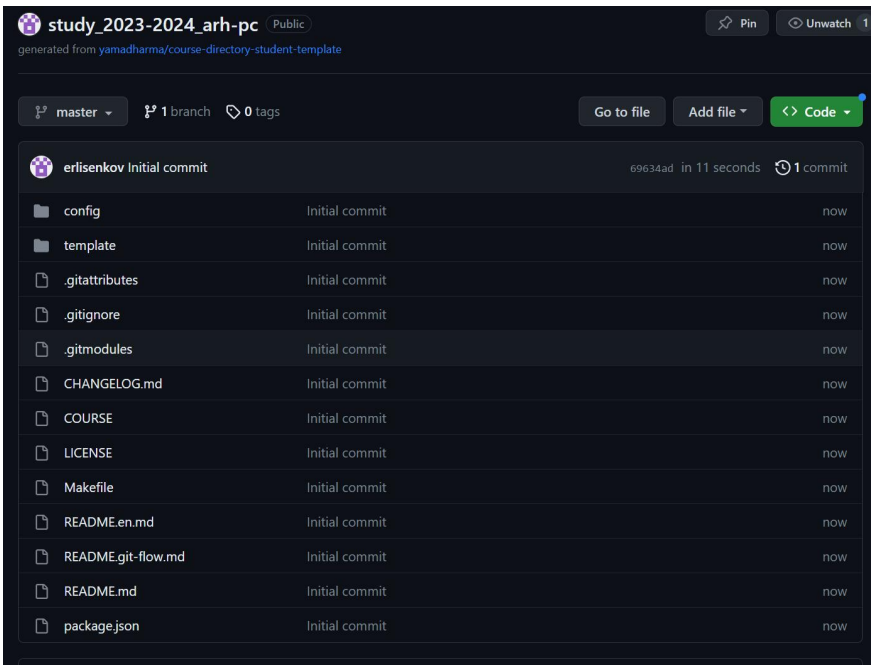


Рис. 4.15 Репозиторий создан

Далее перехожу в созданный каталог курса с помощью cd (рис. 4.16).

```
[erlisenkov@fedora ~]$ cd ~/work/study/2023-2024/"Архитектура компьютера"
[erlisenkov@fedora Архитектура компьютера]$
```

Рис. 4.16 Переход в репозиторий

Клонирую созданный репозиторий с помощью команды

git clone –recursive git@github.com:/study_2022–2023_arh-pc.git arch-pc (рис. 4.17).

```
bash: user_name: Нет такого файла или каталога
[erlisenkov@fedora Архитектура компьютера]$ git clone --recursive git@github.com:er
lisenkov/study_2023-2024_arh-pc.git arch-pc
Клонирование в «arch-pc»...
```

Рис. 4.17 Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория (рис. 4.18).

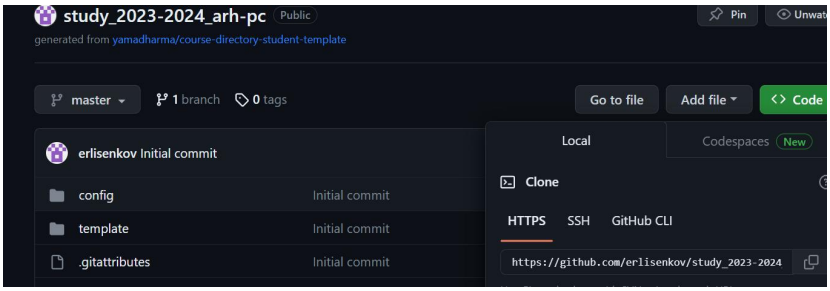


Рис. 4.18 Окно с ссылкой для копирования репозитория

4.6 Настройка каталога курса

Выполню переход в каталог arch-pc с помощью cd (рис. 4.19).

```
[erlisenkov@fedora Архитектура компьютера]$ cd ~/work/study/2023-2024/"Архитектура
компьютера"/arch-pc
[erlisenkov@fedora arch-pc]$
```

Рис. 4.19 Переход по директориям

Удалю лишние файлы благодаря утилите rm (рис. 4.20).

```
[erlisenkov@fedora arch-pc]$ rm package.json
```

Рис. 4.20 Удаление файлов

Создаю необходимые каталоги (рис. 4.21).

```
[erlisenkov@fedora arch-pc]$ echo arch-pc > COURSE
[erlisenkov@fedora arch-pc]$ make
```

Рис. 4.21 Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавил с помощью git add, комментирую, сохраняю изменения и добавляю на сервер с помощью команды git commit (рис. 4.22 - рис. 4.23).

```
[erlisenkov@fedora arch-pc]$ git add .
[erlisenkov@fedora arch-pc]$ git commit -am 'feat(main): make course structure'
[master 40383be] feat(main): make course structure
 199 files changed, 54725 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
 create mode 100644 labs/lab02/presentation/Makefile
 create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab02/presentation/presentation.md
 create mode 100644 labs/lab02/report/Makefile
 create mode 100644 labs/lab02/report/bib/cite.bib
 create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
 create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
 create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
 create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
 create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py
 create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py
 create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/main.py
 create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/pandocattributes.py
 create mode 100644 labs/lab02/report/report.md
```

Рис. 4.22 Загрузка на сервер

```
[erlisenkov@fedora arch-pc]$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.13 КиБ | 1.69 МиБ/с, готово.
Всего 35 (изменений 4), повторно использовано 0 (изменений 0), повторно использован
о пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:erlisenkov/study_2023-2024_arh-pc.git
   69634ad..40383be  master -> master
[erlisenkov@fedora arch-pc]$
```

Рис. 4.23 Выгрузка на сервер

Выполню проверку правильности выполнения работы на GitHub (рис. 4.24)

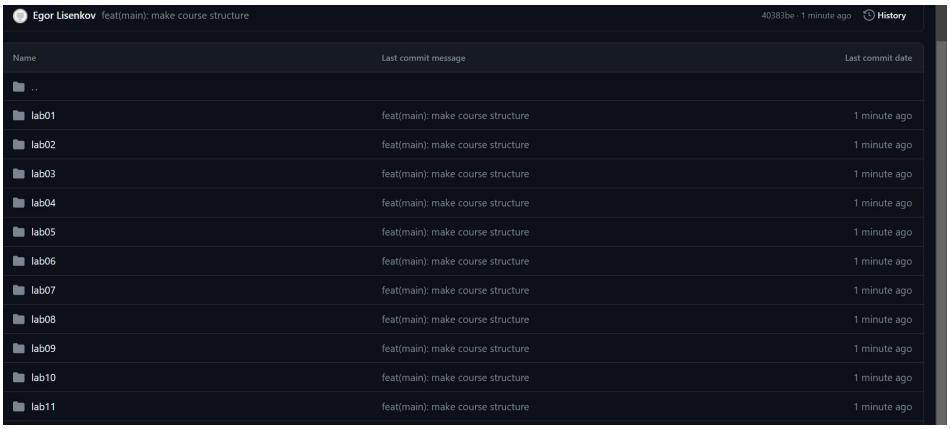


Рис. 4.24 Страница репозитрория

4.7 Выполнение заданий для самостоятельной работы

1) Выполню переход в директорию labs/lab03/report с помощью утилиты cd. Создам файл для отчёта с помощью утилиты touch (рис 4.25).

```
[erlisenkov@fedora report]$ touch ЛО2_Лисенков_отчёт
```

Рис. 4.25 Создание файла

Оформить отчёт я могу в LibreOffice, поэтому открываю пустой лист без имени и в самом файле открываю недавно созданный отчёт в lab03/report (рис. 4.26).

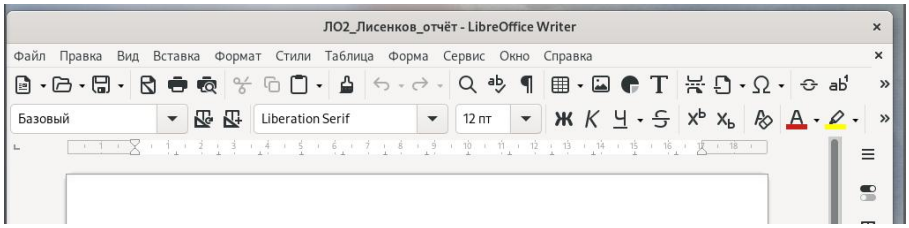


Рис. 4.26 Открытый файл из lab03/report

2) Выполню переход в подкаталог lab01/report (рис. 4.27).

```
[erlisenkov@fedora report]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab01/report
[erlisenkov@fedora report]$ pwd
/home/erlisenkov/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report
```

Рис. 4.27 Переход между каталогами

Проверю местонахождения файлов с отчётами по лабораторным, с помощью команды ls (рис. 4.28).

```
[erlisenkov@fedora arch-pc]$ ls ~/Загрузки
Л01_Лисенков_отчет.pdf  Л02_Лисенков_отчёт.doc
[erlisenkov@fedora arch-pc]$
```

Рис. 4.28 Проверка в директории Загрузки

Копирую первую лабораторную и вторую с помощью утилиты cp и проверяю правильность выполнения команды cp с помощью ls (рис. 4.29).

```
[erlisenkov@fedora report]$ cp ~/Загрузки/Л01_Лисенков_отчет.pdf /work/study/2023-2024/'Архитектура компьютера'/arch-pc/labs/lab01/report
[erlisenkov@fedora report]$ cp ~/Загрузки/Л02_Лисенков_отчёт.doc /home/work/study/2023-2024/'Архитектура компьютера'/arch-pc/labs/lab02/report
```

Рис. 4.29 Копирование файлов

Выполню проверку файлов и пойму, что сделал всё правильно (рис. 4.30).

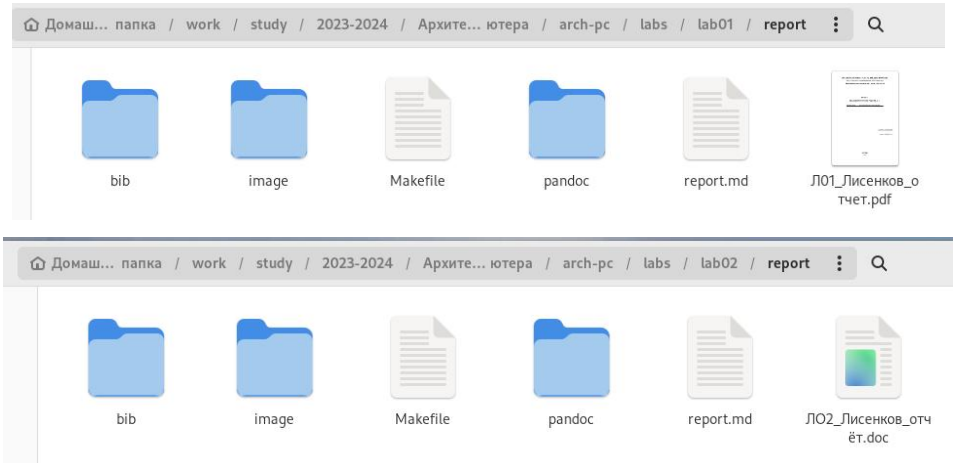


Рис. 4.30 Нахождение файлов

3) Добавляю с помощью команды git add в коммит созданные файлы (рис. 4.31).

```
[erlisenkov@fedora report]$ git add Л02_Лисенков_отчёт.doc
```

Рис. 4.31 Добавление на сервер

Перехожу в директорию к первой лабораторной и тоже добавляю её на сервер (рис. 4.32).

```
[erlisenkov@fedora report]$ git add Л01_Лисенков_отчет.pdf
```

Рис. 4.32 Добавление на сервер 2-ого файла

Сохраняю изменения на сервере и поясняю, что добавил файл (рис. 4.33).

```
[erlisenkov@fedora report]$ git commit -m "Add existing file"
Текущая ветка: master
Эта ветка соответствует «origin/master».

нечего коммитить, нет изменений в рабочем каталоге
[erlisenkov@fedora report]$
```

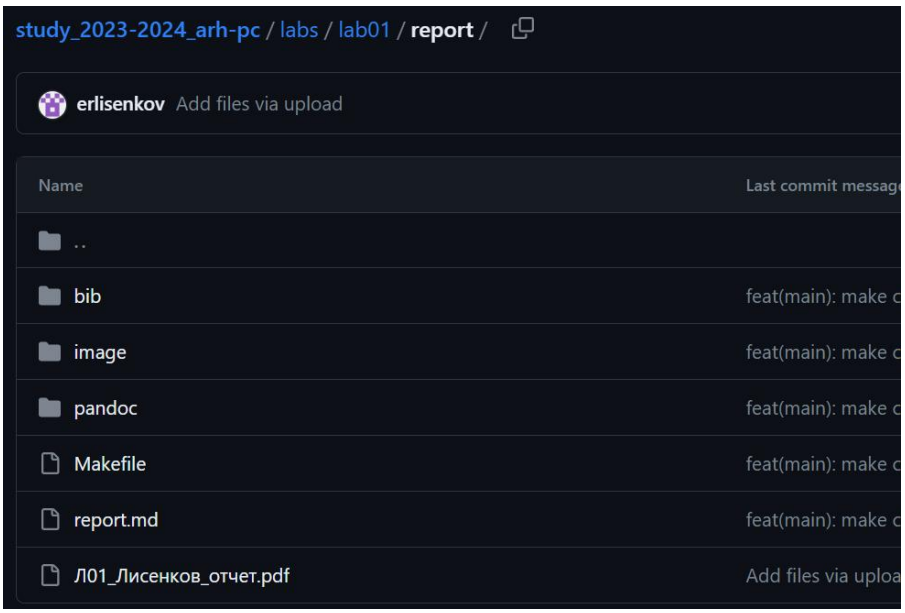
Рис. 4.33 Сохраняю на сервере

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. 4.34).

```
[erlisenkov@fedora report]$ git push -f origin master
Перечисление объектов: 74, готово.
Подсчет объектов: 100% (74/74), готово.
Сжатие объектов: 100% (66/66), готово.
Запись объектов: 100% (74/74), 1.40 МиБ | 2.24 МиБ/с, готово.
Всего 74 (изменений 11), повторно использовано 26 (изменений 1), повтор
но использовано пакетов 0
remote: Resolving deltas: 100% (11/11), done.
To github.com:erlisenkov/study_2023-2024_arh-pc.git
 + 9a6c37f...88cc3a5 master -> master (forced update)
[erlisenkov@fedora report]$
```

Рис. 4.34 Отправка в центральный репозиторий

Проверяю на сайте GitHub правильность выполнения задания (рис. 4.35).



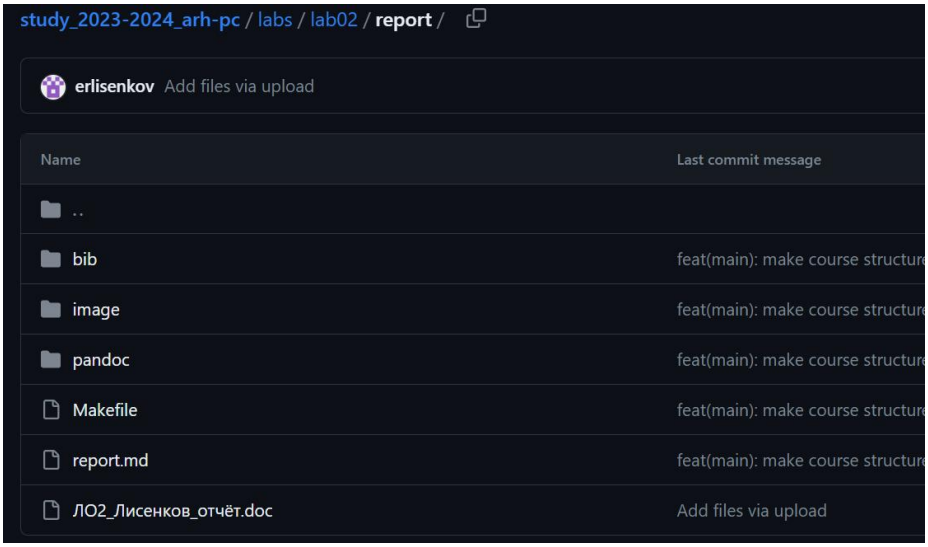


Рис. 4.35 Проверка выполненных команд

5 Вывод

Из сегодняшней лабораторной я понял как нужно работать с системой контроля GitHub. Эти знания понадобятся мне в дальнейшей работе, когда мне придётся работать в команде, где нужен будет контроль над файлами.

6 Источники

Лабораторная_работа_№2_Систем