

Отчёта по лабораторной работе №3

дисциплина: Операционные системы

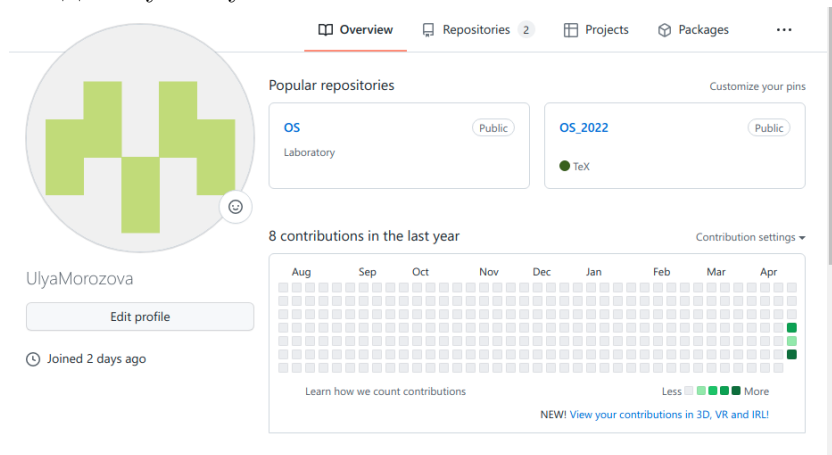
Морозова Ульяна Константиновна

Цель работы

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

Ход работы

1. Создаем учетную запись на GitHub и заполним основные данные (рис.1).



2. Настроим базовую конфигурацию git. Для этого зададим имя и email владельца репозитория (рис.2). Настроим utf-8 в выводе сообщений git. Настроим верификацию и подписание коммитов git: зададим имя начальной ветки (будем называть её master), параметр autocrlf, параметр safecrlf (рис.3).

```
ukmorozova@dk6n50 ~ $ git config --global user.name "Ulyana"
ukmorozova@dk6n50 ~ $ git config --global user.email "morozova.uk03@gmail.com"
```

```
ukmorozova@dk6n50 ~ $ git config --global core.quotepath false
ukmorozova@dk6n50 ~ $ git config --global init.defaultBranch master
ukmorozova@dk6n50 ~ $ git config --global core.autocrlf input
ukmorozova@dk6n50 ~ $ git config --global core.safecrlf warn
```

Рис. 0.1: Рис.3

3. Теперь нам надо сгенерировать для ключа ssh и gpg и вставить их в учетную запись github для того, чтобы привязать наш компьютер с github. Создадим ключ ssh с помощью команды (рис.4)

```
ssh-keygen -t rsa -b 4096
```

и скопируем его в буфер обмена командой (рис.5)

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

```
ukmorofova@dk6n50 ~ $ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/u/k/ukmorofova/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/u/k/ukmorofova/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/u/k/ukmorofova/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/u/k/ukmorofova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tEkIkDfJVx20T7yLcTBGbFL2RATcLe7ndCbvZtcGbbY ukmorofova@dk6n50
The key's randomart image is:
+----[RSA 4096]-----+
| .+.. ..*B*+.. |
| . =...o=o=.. |
| . o. oo=.+. |
| |
| o + =.. |
| S ..+ . |
| +..= * |
| . + Oo |
| .EB |
| =o |
+----[SHA256]-----+
```

Рис. 0.2: Рис.4

```
ukmorofova@dk6n50 ~ $ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 0.3: Рис.5

Теперь вставим ключ в аккаунт на GitHub (рис.9). Создадим ключ gpg командой

```
gpg --full-generate-key
```

и выбираем из предложенных опций варианты, которые указаны в лабораторной работе (рис.6). Выведем список ключей, чтобы скопировать отпечаток приватного ключа (рис.7).

```

ukmorozova@dk6n50 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.27; 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.

Выберите тип ключа:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
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

```

Рис. 0.4: Рис.6

```

ukmorozova@dk6n50 ~ $ gpg --list-secret-keys --keyid-format LONG
/afs/.dk.sci.pfu.edu.ru/home/u/k/ukmorozova/.gnupg/pubring.kbx
-----
sec   rsa4096/12739AE5F41C9B93 2022-04-21 [SC]
      95367C87991996FC037525E712739AE5F41C9B93
uid           [ абсолютно ] UlyaMorozova <morozova.uk03@gmail.com>
ssb   rsa4096/1509596CAB4309DD 2022-04-21 [E]
-----
sec   rsa4096/815735FC9C2E1E0B 2022-04-22 [SC]
      FEB25E6FE9CE7E8189F7BFA9815735FC9C2E1E0B
uid           [ абсолютно ] UlyaMorozova <morozova.uk03@gmail.com>
ssb   rsa4096/B844121380933902 2022-04-22 [E]
-----
sec   rsa4096/6484FFAD28DA85B9 2022-04-22 [SC]
      4A16140E17416823F18E79736484FFAD28DA85B9
uid           [ абсолютно ] Ulyana <morozova.uk03@gmail.com>
ssb   rsa4096/59D356D01D1AEE42 2022-04-22 [E]

```

Рис. 0.5: Рис.7

Скопируем сгенерированный PGP ключ в буфер обмена (рис.8):

```

ukmorozova@dk6n50 ~ $ gpg --armor --export 6484FFAD28DA85B9 | xclip -sel clip
ukmorozova@dk6n50 ~ $ git config --global user.signingkey 6484FFAD28DA85B9
ukmorozova@dk6n50 ~ $ git config --global commit.gpgsign true
ukmorozova@dk6n50 ~ $ git config --global gpg.program $(which gpg2)

```


Рис. 0.6: Рис.8


И вставим его на GitHub (рис.9)

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

**morozova.uk03@gmail.com**
SHA256: krI4rmAIaDBwoXZpUhp82ovFaj382/ALVCa3evMvXY8
SSH Added on 22 Apr 2022
Last used within the last week — Read/write [Delete](#)


**ukmorofova@dk6n50**
SHA256: tEkIkDFjVx20T7yLcTBGbFL2RATcLe7ndCbvZtcGbbY
SSH Added on 22 Apr 2022
Last used within the last week — Read/write [Delete](#)

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

**Email address:** morozova.uk03@gmail.com
Key ID: 6484FFAD28DA85B9
Subkeys: 59D356D01D1AEE42
GPG Added on 22 Apr 2022 [Delete](#)

Learn how to [generate a GPG key and add it to your account](#).

Рис. 0.7: Рис.9

- Используя введенный email, укажем Git применять его при подписи коммитов (рис.10)

```
ukmorofova@dk6n50 ~ $ gpg --armor --export 6484FFAD28DA85B9 | xclip -sel clip
ukmorofova@dk6n50 ~ $ git config --global user.signingkey 6484FFAD28DA85B9
ukmorofova@dk6n50 ~ $ git config --global commit.gpgsign true
ukmorofova@dk6n50 ~ $ git config --global gpg.program $(which gpg2)
```

Рис. 0.8: Рис.10

- Создадим путь, где будут храниться материалы к лабораторным работам и перейдем в последнюю папку (рис.11) и скачаем шаблон репозитория (рис.11) в папку.

```
ukmorozova@dk6n50 ~$ mkdir -p ~/work/study/2021-2022/Операционные системы
ukmorozova@dk6n50 ~$ cd ~/work/study/2021-2022/Операционные системы
ukmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы$ git clone --recursive https://github.com/yamadharma/course-directory-student-template.git
```

Рис. 0.9: Рис.11

Теперь создадим репозиторий на GitHub, где будут храниться только что созданные папки.

6. Перейдем в каталог курса (команда `cd os-intro`), удалим лишние файлы (команда `rm package.json`) и создадим необходимые каталоги для дальнейшей работы (`make COURSE=os-intro`) (рис.12).

```
ukmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/os-intro$ ls
config  LICENSE  project-personal  README.git-flow.md  structure
labs    Makefile  README.en.md      README.md            template
```

Рис. 0.10: Рис.12

Теперь скопируем наш собственный репозиторий OS_2022 в папку “Операционные системы”, перенесем в него все файлы из папки `os-intro` и отправим файлы на сервер, чтобы они также появились в репозитории на GitHub (рис.13-16).

```
ukmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/os-intro$ git clone --recursive git@github.com:UlyaMorozova/OS_2022.git
Клонирование в «OS_2022»...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Получение объектов: 100% (3/3), готово.
```

Рис. 0.11: Рис.13

```
ukmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/os-intro$ cd ..
ukmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/OS_2022$ git add .
```

Рис. 0.12: Рис.14

```

ulkmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/OS_2022 $ git commit -am 'one'
[main ea2707d] one
185 files changed, 20098 insertions(+), 2 deletions(-)
create mode 100644 LICENSE
create mode 100644 Makefile
create mode 100644 README.en.md
create mode 100644 README.git-flow.md
create mode 100644 config/course/os-intro
create mode 100644 config/course/sciprog-intro
create mode 100755 config/script/lab
create mode 100755 config/script/project-group
create mode 100755 config/script/project-personal
create mode 100644 labs/lab01/presentation/Makefile
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 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
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 100644 labs/lab02/report/report.md
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/presentation/Makefile
create mode 100644 labs/lab04/presentation/presentation.md
create mode 100644 labs/lab04/report/Makefile
create mode 100644 labs/lab04/report/bib/cite.bib
create mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg

```

Рис. 0.13: Рис.15

```

ulkmorozova@dk6n50 ~/work/study/2021-2022/Операционные системы/OS_2022 $ git push
Перечисление объектов: 51, готово.
Подсчет объектов: 100% (51/51), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (45/45), готово.
Запись объектов: 100% (49/49), 446.18 Киб | 3.54 Миб/с, готово.
Всего 49 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), done.
To github.com:UlyaMorozova/OS_2022.git
 9f31c87..ea2707d main -> main

```

Рис. 0.14: Рис.16

Смотрим репозиторий на сайте GitHub и убеждаемся, что мы сделали все правильно(рис.17).

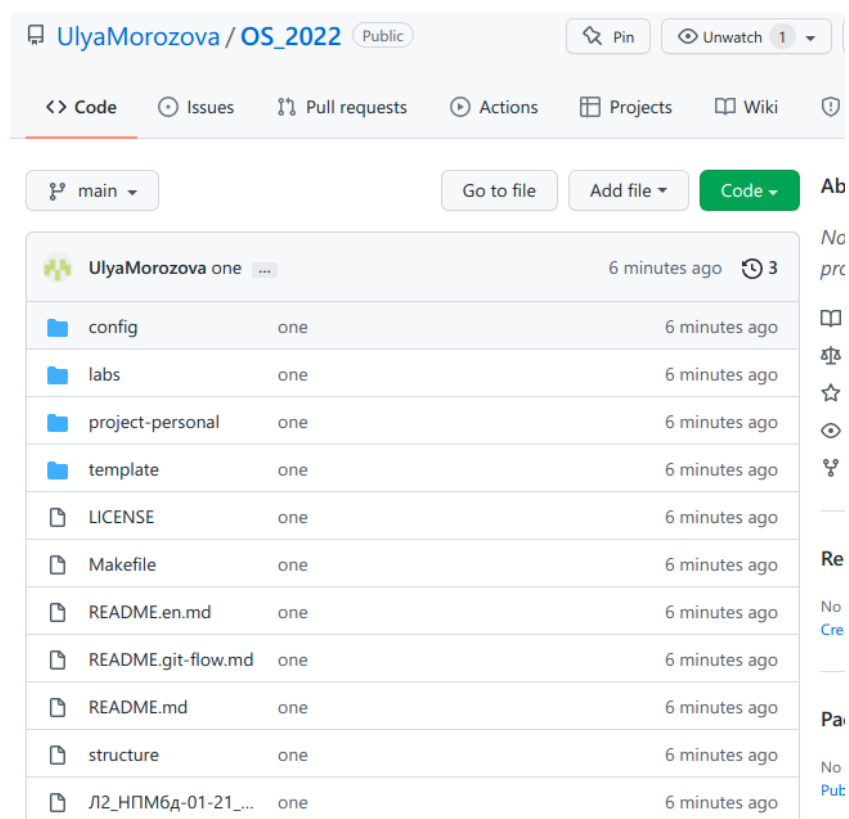


Рис. 0.15: Рис.17

Выводы

Я изучила идеологию и применения средств контроля версий и освоила умения по работе с git.

#Контрольные вопросы

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. 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. 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.
4. 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 дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

5. 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.
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 директо-

рии): `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 стекущим 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. Исползования `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`:

1. нужно постоянно создавать архивы с рабочим кодом
2. сложно “переключаться” между архивами
3. сложно перетаскивать изменения между архивами
4. легко что-то напутать или потерять

10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.