

Лабораторная работа №1

Научное программирование

Дарижапов Тимур Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	12
5	Ответы на контрольные вопросы	13
6	Список литературы	17

List of Tables

List of Figures

3.1	Учетная запись	7
3.2	Идентификация и создание ключа	8
3.3	Создание ключа	8
3.4	Создание ключа pgr	9
3.5	Настройка ключа	9
3.6	Список и отпечаток	9
3.7	Демонстрация ключа	10
3.8	Отображение ключа	10
3.9	Отображение ключа	11
3.10	Отображение ключа	11

1 Цель работы

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

2 Задание

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

3 Выполнение лабораторной работы

Создаём учётную запись на <https://github.com>.



Figure 3.1: Учетная запись

Настроим систему контроля версий git, как это описано в лабораторной работе с использованием сервера репозитория <https://github.com/> 1) Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"`. Настроим верификацию и подписание коммитов git. Зададим имя начальной ветки. 2) Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняться в каталоге `~/.ssh/`. Генерируем ключ двумя способами.


```
timur@tadarizhapov:~/InfoBez$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
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 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
```

Figure 3.4: Создание ключа gpg

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: tadarizhapov
Адрес электронной почты: timaanddar@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "tadarizhapov <timaanddar@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печатать
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 (печатать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: сертификат отзёма записан в '/home/timur/.gnupg/openpgp-revocs.d/3D143658FB71129C1C77EEE4F3917C7F999515CA.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-09-14 [SC]
       3D143658FB71129C1C77EEE4F3917C7F999515CA
uid    tadarizhapov <timaanddar@mail.ru>
sub    rsa4096 2024-09-14 [E]
```

Figure 3.5: Настройка ключа

Выводим список ключей и копируем отпечаток приватного ключа.

```
timur@tadarizhapov:~/InfoBez$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: gpg
gpg: глубина: 0 достоверных: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 2u
/home/timur/.gnupg/pubring.kbx
-----
sec   rsa1024/2B3339CA12873B53B 2024-09-14 [SC]
       63E035958755E9DEE474A9112B339CA12873B53B
uid    [ абсолютно ] Timur <tidaan@mail.ru>
ssb    rsa1024/40D9E73D131421C4 2024-09-14 [E]

sec   rsa4096/F3917C7F999515CA 2024-09-14 [SC]
       3D143658FB71129C1C77EEE4F3917C7F999515CA
uid    [ абсолютно ] tadarizhapov <timaanddar@mail.ru>
ssb    rsa4096/D88B56BBEB627544 2024-09-14 [E]

timur@tadarizhapov:~/InfoBez$
```

Figure 3.6: Список и отпечаток

Скопируем наш сгенерированный PGP ключ в буфер обмена.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mI0EZuWpogEEAL0JQ0xbD20I9ETg6UXQN6lUdbi/v8V3weET8phb9P9X82KN2Di6  
q32onugbfS4zNR+goiOKJp6LbkvnHiD4hBBFC7zXg/JpUdy703qdnYGVXkwlhi/  
iNu/UNT+lFWceo9Zta5r4QzMaYtxxNb4axmN0mj8heRfkSy3pc8XSLvxABEBAAG0  
FLRpbXVyIDx0aWRhYVW5AbWFpbC5ydT6I0QQTAQoA0xYhBGPgNZWHVene5HSpESsz  
nKEoc7U7BQJm5amiAhsDBQsJCACCAiICBhUKCQgLAgQWAgMBAh4HAheAAAJECsz  
nKEoc7U7YyQD/0JY4eeTAv1+BdH4z529hPpi1ya8dFG3S1spMQwcE+pcc1mwMUWe  
9vYzgBnL5Pef2E8Bg0DqBDR8YR7wleNILJzIstWbUAxFPSFjmyuzP3xoy/hGEMlr  
v2I6DaRvR3X0FstJ2/4H1e1FUzDp/Ozqbv1k0LwNgLapdiK+ju5GBQY0uI0EZuWp  
ogEEAONzxa5qKhrF4DXBviuqtI5cTCuTmfuP0vP1xYamppmuPPL47gSzode3znGu  
1n7DC0v4PP0QAlklN9JvLwRmW3L8sCsVKUIgH6sgMZwx9GaxFEqRpgR5zVngrM/d  
iH+B2eDQpYZ3sUDN0rjMRIRvW9qmHmEez7/1NWMDKoLwthI9ABEBAAGItgQYAQoA  
IBYhBGPgNZWHVene5HSpESsznKEoc7U7BQJm5amiAhsMAAJECsznKEoc7U76NsD  
/iMuESE2DSzAPbfM+uCctQLuF2RrucDu0YnuGRorhQTX1hrtD0g9d89YYP10U0l1  
MEEm09utqD0IJGue3zUeIFiZY03Jt/GYNzoFCNRsZrHuBpKmqJRKt5gm/jbZXnjB  
bHaKG7LS0KwMh4Yfeg4ZhaYIBw1/9qR79E4VlrUaAKXX  
=Boi3  
-----END PGP PUBLIC KEY BLOCK-----  
timur@tadarizhapov: ~/InfoBez$
```

Figure 3.7: Демонстрация ключа

Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

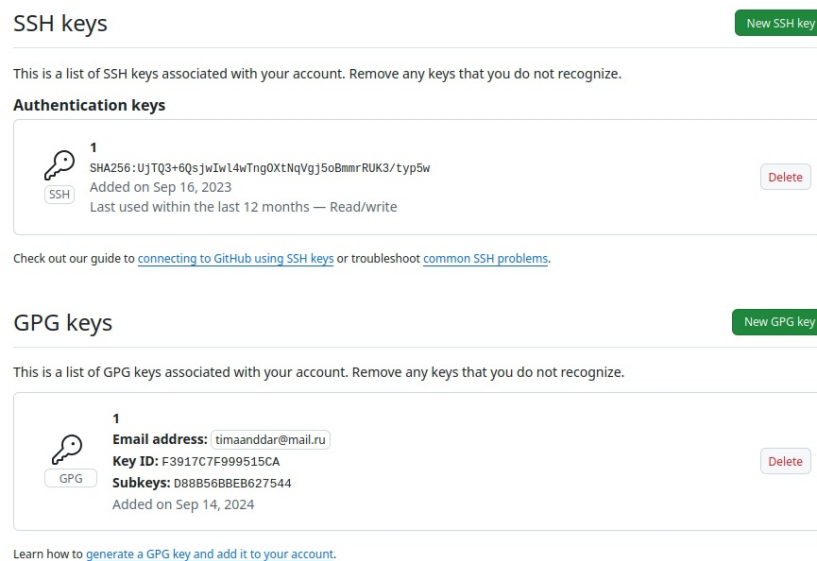


Figure 3.8: Отображение ключа

Осталось настроить автоматические подписи с помощью команд:

```
timur@tadarizhapov:~/InfoBez$ git config --global user.signingkey F3917C7F999515CA
timur@tadarizhapov:~/InfoBez$ git config --global commit.gpgsign true
timur@tadarizhapov:~/InfoBez$ git config --global gpg.program $(which gpg2)
bash: which gpg2: синтаксическая ошибка в выражении (неверный маркер «gpg2»)
timur@tadarizhapov:~/InfoBez$ git config --global gpg.program $(which gpg2)
timur@tadarizhapov:~/InfoBez$ mkdir -p ~/work/2024-2025/'Научное программирование'
timur@tadarizhapov:~/InfoBez$ cd ~/work/2024-2025/'Научное программирование'
timur@tadarizhapov:~/work/2024-2025/Научное программирование$
```

Figure 3.9: Отображение ключа

Далее мы создаём репозиторий на основе шаблона и отправляем все имеющиеся данные на GitHub.

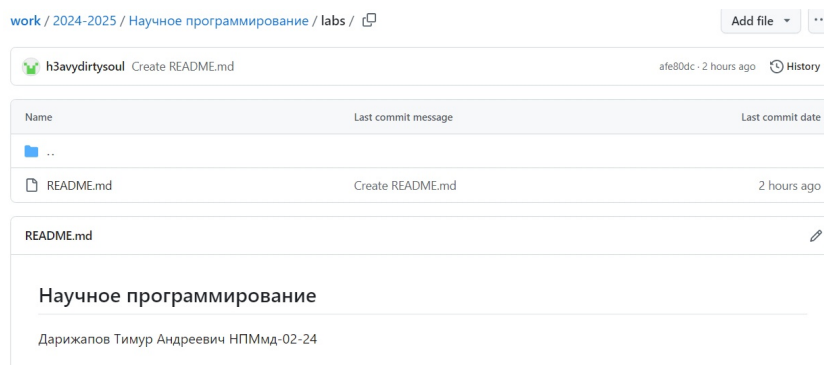


Figure 3.10: Отображение ключа

4 Выводы

В ходе выполнения лабораторной работы я изучил идеологию и применение средств контроля версий, а также освоил умения по работе с git.

5 Ответы на контрольные вопросы

1) Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

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

3) 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.

4) 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`. После этого в каталог `etutorial` появится каталог `.git`, в котором будет храниться история изменений. Создадим тестовый текстовый файл `hello.txt` и добавим его в локальный репозиторий. `echo 'hello world' > hello.txt git add hello.txt git commit -am 'Новый файл'`. Воспользуемся командой `status` для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии: `git status`.

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/`. Существует несколько доступных серверов репозитория с возможностью бесплатного размещения данных. Например, <https://github.com/>. Для работы с ним необходимо сначала зайти на сайт <https://github.com/> и создать учётную запись. Затем необходимо загрузить сгенерённый нами ранее открытый ключ. Для этого зайти на сайт <https://github.com/> под своей учётной записью и перейти в меню `GitHub setting`. После этого выбрать в боковом меню `GitHub setting` SSH-ключи и нажать кнопку `Добавить ключ`. Скопировав из локальной консоли ключ в буфер обмена: `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле. После этого можно создать на сайте репозиторий, выбрав в меню `Репозитории` `Создать репозиторий`, дать ему название и сделать общедоступным (публичным). Для загрузки репозитория из локального каталога на сервер выполняем следующие команды: `git remote add origin ssh://git@github.com:/.git` `git push -u origin master`. Далее на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии

центрального репозитория.

6) У Git две основных задачи: первая — хранить информацию о всех изменениях в коде, а вторая — обеспечение удобства командной работы над кодом.

7) 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 файлы/или каталоги из индекса репозитория (при этом файлы/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 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) Гид для создания текстового файла `echo 'hello world' > hello.txt` `git add hello.txt`

9) Для фиксации истории проекта в рамках этого процесса вместо одной ветки `master` используются две ветки. В ветке `master` хранится официальная история релиза, а ветка `develop` предназначена для объединения всех функций. Ветви

решают следующие проблемы нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко запутаться в файлах

- 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. Для этого сначала нужно получить список 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`.

6 Список литературы

Лабораторная работа №1

Лабораторная работа № 2. Управление версиями [Электронный ресурс]. 2019.

URL:<https://esystem.rudn.ru/mod/resource/view.php?id=1154997>