

отчёт по лабораторной работе №2”

Управление версиями

Чванова Ангелина Дмитриевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
3.1	Системы контроля версий. Общие понятия	6
4	Выполнение лабораторной работы	8
5	Выводы	16
6	Контрольные вопросы	17

Список иллюстраций

4.1	терминал с введенными командами для установки git-flow	8
4.2	терминал с введенными командами для установки git-flow	9
4.3	терминал с введенной командой для установки gh	9
4.4	Ввод имени владельца и email , а также настройка utf-8	10
4.5	Выбор имени начальной ветки и настройка параметров safecrlf и autocrlf	10
4.6	Создание ключа ssh по алгоритму rsa с размером для ключа 4096 бит в консоли	11
4.7	консоль с введенной командой для создания ключей pgr и предложенные опции	12
4.8	выбранные опции для создания ключей pgr	12
4.9	Сгенерированный в консоли ключ	12
4.10	Копирование в консоле сгенерированного PGP ключа в буфер обмена	13
4.11	вставка полученного ключа в поле ввода (переход в настройки GitHub (https://github.com/settings/keys)	13
4.12	Авторизация в gh	14
4.13	удаление лишних файлов, создание каталогов и отправка файлов на сервер	15

1 Цель работы

изучить идеологию и применение средств контроля версий, а также освоить умения по работе с git.

2 Задание

– Создать базовую конфигурацию для работы с git. – Создать ключ SSH. – Создать ключ PGP. – Настроить подписи git. – Зарегистрироваться на Github. – Создать локальный каталог для выполнения заданий по предмету

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 дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных

В табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

1. Установка git-flow в Fedora Linux

Установка проходила вручную с помощью данных команд(рис. 4.1 -4.2)

```
cd /tmp
```

```
wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes
```

```
/gitflow/develop/contrib/gitflow-installer.sh
```

```
chmod +x gitflow-installer.sh
```

```
sudo ./gitflow-installer.sh install stable
```

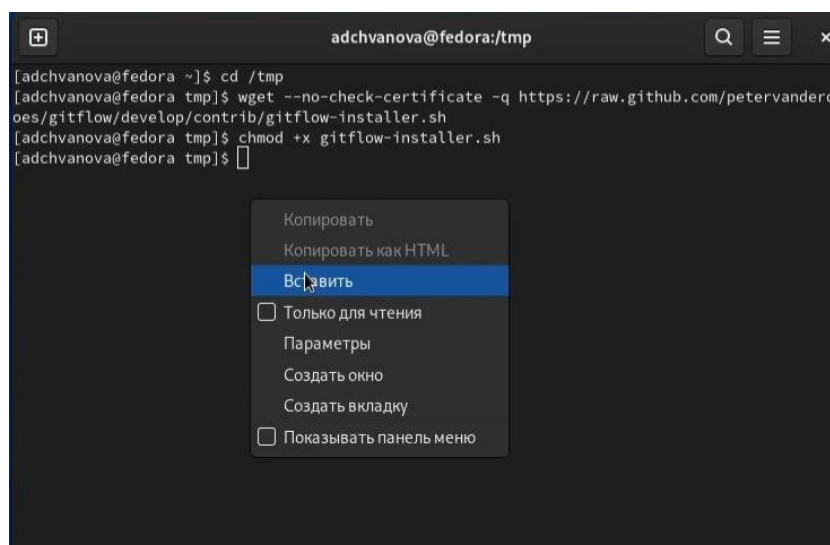


Рис. 4.1: терминал с введенными командами для установки git-flow


```
adchvanova@fedora/tmp — sudo ./gitflow-installer.sh install stable
[adchvanova@fedora ~]$ cd /tmp
[adchvanova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[adchvanova@fedora tmp]$ chmod +x gitflow-installer.sh
[adchvanova@fedora tmp]$ sudo ./gitflow-installer.sh install stable

Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

№1) Уважайте частную жизнь других.
№2) Думайте, прежде что-то вводить.
№3) С большой властью приходит большая ответственность.

[sudo] пароль для adchvanova:
### git-flow no-make installer ###
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
```

Рис. 4.2: терминал с введенными командами для установки git-flow

2. Установка gh в Fedora Linux с помощью команды sudo dnf install gh(рис4.3)

```
adchvanova@fedora/tmp — sudo dnf install gh
'lease-publish'
'gitflow/hooks/pre-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-start'
'gitflow/hooks/pre-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-track'
[adchvanova@fedora tmp]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:24:00 назад, Чт 21 апр 2022 21:48:12.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.7.0-1.fc35 updates      6.8 М

Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 6.8 М
Объем изменений: 32 М
Продолжить? [д/Н]: д
Загрузка пакетов:
gh-2.7.0-1.fc35.x86_64.rpm 43% [=====] 601 kB/s | 3.0 MB 00:06 ETA
```

Рис. 4.3: терминал с введенной командой для установки gh

3. Базовая настройка git(рис 4.4-4.5)

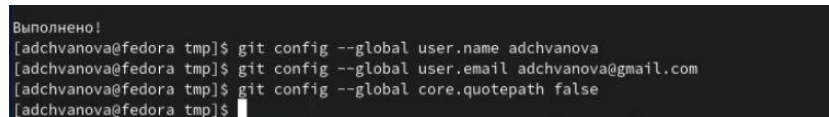
– Задаем имя и email владельца репозитория с помощью команд:

git config --global user.name "Name Surname"

git config --global user.email "work@mail"

Настройка utf-8 в выводе сообщений git:

`git config --global core.quotepath false`



```
Выполнено!  
[adchvanova@fedora tmp]$ git config --global user.name adchvanova  
[adchvanova@fedora tmp]$ git config --global user.email adchvanova@gmail.com  
[adchvanova@fedora tmp]$ git config --global core.quotepath false  
[adchvanova@fedora tmp]$
```

Рис. 4.4: Ввод имени владельца и email , а также настройка utf-8

Настройте верификацию и подписание коммитов git.

– Задаем имя начальной ветки (будем называть её master):

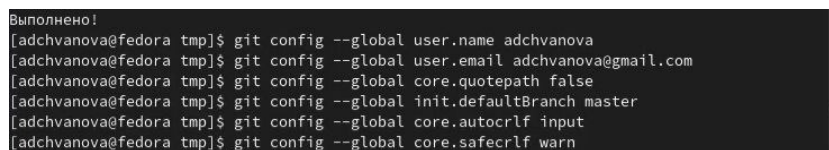
`git config --global init.defaultBranch master`

– Параметр autocrlf:

`git config --global core.autocrlf input`

– Параметр safecrlf:

`git config --global core.safecrlf warn`



```
Выполнено!  
[adchvanova@fedora tmp]$ git config --global user.name adchvanova  
[adchvanova@fedora tmp]$ git config --global user.email adchvanova@gmail.com  
[adchvanova@fedora tmp]$ git config --global core.quotepath false  
[adchvanova@fedora tmp]$ git config --global init.defaultBranch master  
[adchvanova@fedora tmp]$ git config --global core.autocrlf input  
[adchvanova@fedora tmp]$ git config --global core.safecrlf warn
```

Рис. 4.5: Выбор имени начальной ветки и настройка параметров safecrlf и autocrlf

4.Создание ключей ssh(рис4.6)

Ключ создавался по алгоритму rsa с размером для ключа 4096 бит:

`ssh-keygen -t rsa -b 4096`

```
[adchvanova@fedora tmp]$ git config --global core.safecrlf warn
[adchvanova@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adchvanova/.ssh/id_rsa):
Created directory '/home/adchvanova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adchvanova/.ssh/id_rsa
Your public key has been saved in /home/adchvanova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:drqRyh0p0N8rDFTxU2Y2R2LSUCu3XGalyJXpChWwE adchvanova@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|      .++oX+      |
|      o.= X =     |
|      + E = +     |
|      o.@ = * .   |
|      .S+.% * o   |
|      . =. + o    |
|      . o.        |
|      . o.        |
|      .+o.        |
+---[SHA256]-----+
[adchvanova@fedora tmp]$
```

Рис. 4.6: Создание ключа ssh по алгоритму rsa с размером для ключа 4096 бит в консоли

5.Создание ключей pgr(рис4.7-4.8)

– Генерируем ключ

gpg –full-generate-key

– Из предложенных опций (рис4.7) выбираем:

– тип RSA and RSA;

– размер 4096;

– выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда).

– GPG запросит личную информацию, которая сохранится в ключе:

– Имя (не менее 5 символов).

– Адрес электронной почты.

– При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.

– Комментарий.

```

(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.

```

Рис. 4.7: консоль с введенной командой для создания ключей pgp и предложенные опции

```

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: adchvanova
Адрес электронной почты: adchvanova@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "adchvanova <adchvanova@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?

```

Рис. 4.8: выбранные опции для создания ключей pgp

6.Добавление PGP ключа в GitHub(рис4.9,4.10,4.11)

– Выводим список ключей и копируем отпечаток приватного ключа: `gpg --list-secret-keys --keyid-format LONG`

– Формат строки (рис 4.9):

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до] ID_ключа

```

sec  rsa4096/93DDF3A55158768F 2022-04-21 [SC]
    D47BBD4A856F85E430C6DD9893DDF3A55158768F
uid      [ абсолютно ] adchvanova <adchvanova@gmail.com>
ssb  rsa4096/C54F75CBDFFD002 2022-04-21 [E]

```

Рис. 4.9: Сгенерированный в консоли ключ

– Копирование сгенерированного PGP ключа в буфер обмена(рис4.10):

`gpg --armor --export | xclip -sel clip`

```
[adchvanova@fedora tmp]$ gpg --armor --export 93DDF3A55158768F | xclip -sel clip
```

Рис. 4.10: Копирование в консоле сгенерированного PGP ключа в буфер обмена

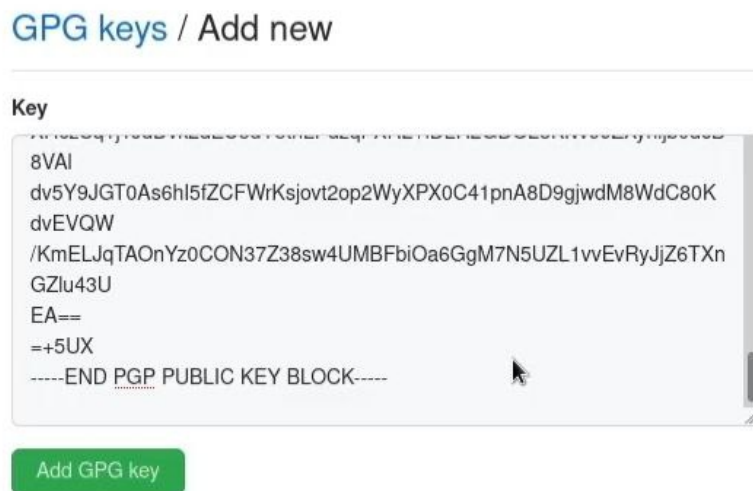


Рис. 4.11: вставка полученного ключа в поле ввода (переход в настройки GitHub (<https://github.com/settings/keys>))

7.Настройка автоматических подписей коммитов git (рис ??)

– Используя введённый email, указываем Git применять его при подписи коммитов:

`git config --global user.signingkey`

`git config --global commit.gpgsign true`

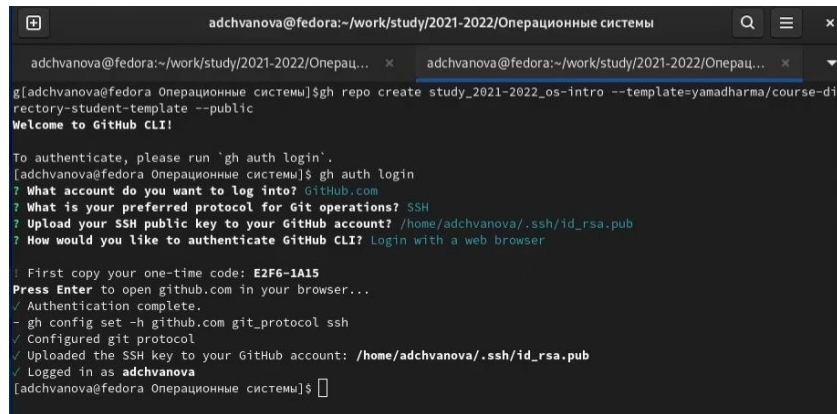
`git config --global gpg.program $(which gpg2)`

```
[adchvanova@fedora tmp]$ git config --global user.signingkey 93DDF3A55158768F
[adchvanova@fedora tmp]$ git config --global commit.gpgsign true
[adchvanova@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

8.Настройка gh(рис4.12)

–команда для авторизации

`gh auth login`



```
adchvanova@fedora:~/work/study/2021-2022/Операционные системы
adchvanova@fedora:~/work/study/2021-2022/Операц... x adchvanova@fedora:~/work/study/2021-2022/Операц... x
g[adchvanova@fedora Операционные системы]$gh repo create study_2021-2022_os-intro --template=yamadharma/course-di
rectory-student-template --public
Welcome to GitHub CLI!

To authenticate, please run 'gh auth login'.
[adchvanova@fedora Операционные системы]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/adchvanova/.ssh/id_rsa.pub
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: E2F6-1A15
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/adchvanova/.ssh/id_rsa.pub
✓ Logged in as adchvanova
[adchvanova@fedora Операционные системы]$
```

Рис. 4.12: Авторизация в gh

9.Создание репозитория курса на основе шаблона и настройка каталога курса
(рис 4.13)

–создание шаблона рабочего пространства.

`mkdir -p ~/work/study/2021-2022/“Операционные системы”`

`cd ~/work/study/2021-2022/“Операционные системы”`

`gh repo create study_2021-2022_os-intro`

☒ `--template=yamadharma/course-directory-student-template --public`

`git clone --recursive`

☒ `git@github.com:/study_2021-2022_os-intro.git os-intro`

– Переход в каталог курса:

`cd ~/work/study/2021-2022/“Операционные системы”/os-intro`

– Удаление лишних файлов:

`rm package.json`

– Создайте необходимые каталоги:

`make COURSE=os-intro`

– Отправьте файлы на сервер:

`git add .`

`git commit -am ‘feat(main): make course structure’`

`git push`

```
adchvanova@fedora:~/work/study/2021-2022/Операционные системы/os-intro — git commit -am feat(main): make...
adchvanova@fedora:~/work/study/2021-2022/Операционн...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (20/20), 12.49 КиБ | 3.12 МБ/с, готово.
Определение изменений: 100% (2/2), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-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/adchvanova/work/study/2021-2022/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 0), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 КиБ | 798.00 КиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/home/adchvanova/work/study/2021-2022/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КиБ | 692.00 КиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eae6bb7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
[adchvanova@fedora Операционные системы]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
[adchvanova@fedora os-intro]$ rm package.json
[adchvanova@fedora os-intro]$ make COURSE=os-intro
[adchvanova@fedora os-intro]$ git add
[adchvanova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 4.13: удаление лишних файлов, создание каталогов и отправка файлов на сервер

5 Выводы

были изучены некоторые команды в консоли для работы с GitHub и применены средства контроля версий, а также освоены умения по работе с git. Удалось создать репозиторий курса на основе шаблона и настроить его. Файлы были отправлены на сервер.

6 Контрольные вопросы

1.Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? 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 – место хранения 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. Commit позволяет отправлять изменения на сервер VCS. История — список всех изменений проекта с возможностью отката в любую точку истории. Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, изменённых.

3.Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Обратите внимание, что нет единого объекта, который получает и отвечает на запрос.

Bitcoin. Давайте возьмем биткойны, например, потому что это самый популярный пример использования децентрализованных систем. Ни одна организация / организация не владеет сетью биткойнов. Сеть представляет собой сумму всех узлов, которые общаются друг с другом для поддержания количества биткойнов, которое есть у каждого владельца счета. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Это наиболее часто используемый тип систем во многих организациях, где клиент отправляет запрос на сервер компании и получает ответ. Wikipedia. Рассмотрим огромный сервер, на который мы отправляем наши запросы, и сервер отвечает запрашиваемой статьей. Предположим, мы ввели поисковый запрос «нездоровая пища» в строке поиска Википедии. Этот поисковый запрос отправляется как запрос на серверы Википедии (в основном, расположенные в штате Вирджиния, США), которые затем возвращают статьи, основанные на релевантности. В этой ситуации мы являемся клиентским узлом, серверы Википедии являются центральным сервером.

4.Опишите действия с VCS при единоличной работе с хранилищем. 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 сообщенийgit:

```
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. Опишите порядок работы с общим хранилищем VCS. 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. Каковы основные задачи, решаемые инструментальным средством git? 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. А также ведение истории изменений, фиксирование изменений, совмещение версий, веток, откат к прошлым версиям.

7. Назовите и дайте краткую характеристику командам git.

`git init` - инициализирует локальный репозиторий

`git add *` или `add.` - добавляет файлы в репозиторий

`git commit` - версия фиксации

`git pull` - загружает текущую версию проекта

`git push` - отправляет измененный проект на сервер

`git checkout` - позволяет переключаться между ветками

`git status` - текущий статус проекта

`git branch` - просмотреть доступные ветки

`git remote add` - добавить удаленный репозиторий

8. Приведите примеры использования при работе с локальным и удалённым репозиториями. Использование git с локальными репозиториями используется для добавления, например, текстового файла в локальный репозиторий

```
git add file.txt
```

```
git commit -am ФАЙЛ
```

9. Что такое и зачем могут быть нужны ветви (branches)? Ветви необходимы, чтобы иметь возможность “разделять” части работы и работать отдельно над каждой имплементацией. Использование ветвей дает возможность обрабатывать нововведения в основную ветвь, которая чаще всего является релизной.
 10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы при commit можно с помощью .gitignore файла. В нем указываются пути, названия, расширения и другие идентификации нежелательных объектов которые не будут учитываться в commit. Это полезно для исключения как “мусорных” файлов, которые не являются значимой частью проекта, а также конфиденциальных файлов, которые содержат в себе приватную информацию, такую как пароли и токены.
1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.
 2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
 3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
 4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
 5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
 6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.