

Отчет по лабораторной работе №2

Операционные системы

Шихалиева Зурият Арсеновна

Содержание

Цель работы	4
Задание	4
Выполнение лабораторной работы. Установка ПО.	4
Выполнение лабораторной работы. Базовая настройка git	5
Выполнение лабораторной работы. Базовая настройка git	5
Выполнение лабораторной работы. Базовая настройка git	5
Выполнение лабораторной работы. Базовая настройка git	5
Выполнение лабораторной работы. Создание ключа SSH	6
Выполнение лабораторной работы. Создание ключа SSH	6
Выполнение лабораторной работы. Создание ключа GPG	7
Выполнение лабораторной работы. Регистрация на Github	8
Выполнение лабораторной работы. Добавление ключа GPG в Github	9
Выполнение лабораторной работы. Добавление ключа GPG в Github	9
Выполнение лабораторной работы. Добавление ключа GPG в Github	9
Выполнение лабораторной работы. Настроить подписи Git	10
Выполнение лабораторной работы. Настройка gh	10
Выполнение лабораторной работы. Настройка gh	11
Выполнение лабораторной работы. Создание репозитория курса на основе шаблона	11
Выполнение лабораторной работы. Создание репозитория курса на основе шаблона	12
Выполнение лабораторной работы. Создание репозитория курса на основе шаблона	12
Выполнение лабораторной работы. Создание репозитория курса на основе шаблона	13
Выводы	13
Контрольные вопросы	13

Список иллюстраций

1	Установка git и gh	4
2	Задаю имя и email владельца репозитория	5
3	Настройка utf-8 в выводе сообщений git	5
4	Задаю имя начальной ветки	5
5	Задаю параметры autocrlf	5
6	Задаю параметры safecrlf	6
7	Генерация ssh ключа по алгоритму rsa	6
8	Генерация ssh ключа по алгоритму ed25519	7
9	Генерация ключа	8
10	Аккаунт на Github	8
11	Вывод списка ключей	9
12	Копирование ключа в буфер обмена	9
13	Добавление нового PGP ключа	10
14	Настройка подписей Git	10
15	Авторизация в gh	11
16	Завершение авторизации	11
17	Создание репозитория	12
18	Удаление файлов и создание каталогов	12
19	Отправка файлов на сервер	13
20	Отправка файлов на сервер	13

Список таблиц

Цель работы

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

Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Настройка gh
6. Создание репозитория курса на основе шаблона.

Выполнение лабораторной работы. Установка ПО.

Устанавливаю необходимое программное обеспечение git и gh (рис. 1).

```
root@zashikhalieva:~# dnf install git gh
Последняя проверка окончания срока действия метаданных: 0:05:34 назад, Пн 26 авг 2024 11:04:59.
Пакет git-2.46.0-1.fc40.x86_64 уже установлен.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.45.0-1.fc40  fedora       8.7 М
```

Рис. 1: Установка git и gh

Выполнение лабораторной работы. Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис. 2).

```
root@zashikhalieva:~# git config --global user.name "Zuriyat Shikhalieva"  
root@zashikhalieva:~# git config --global user.email "yulia.arsenova@mail.ru"
```

Рис. 2: Задаю имя и email владельца репозитория

Выполнение лабораторной работы. Базовая настройка git

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис. 3).

```
root@zashikhalieva:~# git config --global core.quotepath false
```

Рис. 3: Настройка utf-8 в выводе сообщений git

Выполнение лабораторной работы. Базовая настройка git

Начальной ветке задаю имя master (рис. 4).

```
root@zashikhalieva:~# git config --global init.defaultBranch master
```

Рис. 4: Задаю имя начальной ветки

Выполнение лабораторной работы. Базовая настройка git

Задаю параметры autocrlf (рис. 5) и safecrlf (рис. 6).

```
root@zashikhalieva:~# git config --global core.autocrlf input
```

Рис. 5: Задаю параметры autocrlf

```
root@zashikhalieva:~# git config --global core.safecrlf warn
```

Рис. 6: Задаю параметры safecrlf

Выполнение лабораторной работы. Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. 7).

```
root@zashikhalieva:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:nBTXLTkuikctL0+d10MEH/VED6jygTMwIiLVp6cOLZY root@zashikhalieva
The key's randomart image is:
+---[RSA 4096]-----+
| ... . . . 00.0|
|o . o + oo +..+|
|.. . + o.o.+.. o|
| . .o=..++ . |
| o o .S*..= |
| E o o oo.+ . |
| . + o +. o . |
| o +.. |
| . o. |
+----[SHA256]-----+
```

Рис. 7: Генерация ssh ключа по алгоритму rsa

Выполнение лабораторной работы. Создание ключа SSH

Создаю ключ ssh по алгоритму ed25519 (рис. 8).

```

root@zashikhalieva:~# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:oHeU1cRlPkbC06W5Xoofve9EWpH0RQHvrmCDX1Xlm6U root@zashikhalieva
The key's randomart image is:
+--[ED25519 256]--+
|      .Bo+**@|
|      o. O=o=+|
|      . o . =+++|
|      . o ..+++|
|      . . S   E=+|
|      . .   o B |
|      . = o|
|      . o.|
|      .o+|
+-----[SHA256]-----+

```

Рис. 8: Генерация ssh ключа по алгоритму ed25519

Выполнение лабораторной работы. Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. 9).

```

zashikhalieva@zashikhalieva:~$ 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.

gpg: создан каталог '/home/zashikhalieva/.gnupg'
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

```

Рис. 9: Генерация ключа

Выполнение лабораторной работы. Регистрация на Github

Мой аккаунт на GitHub (рис. 10).

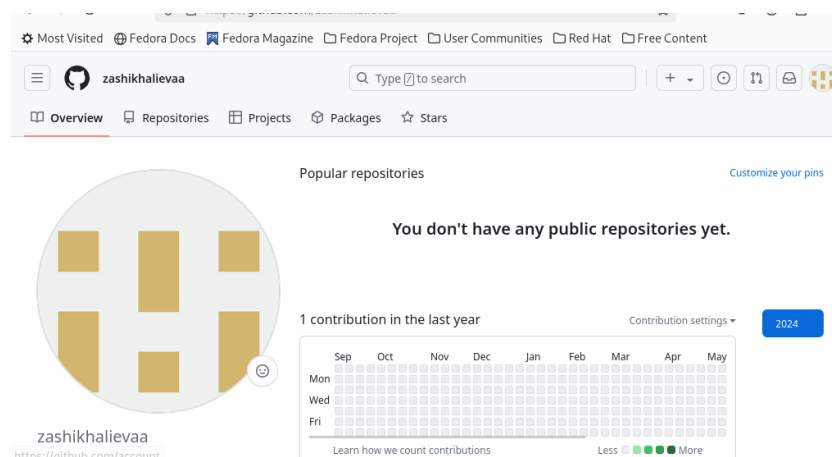
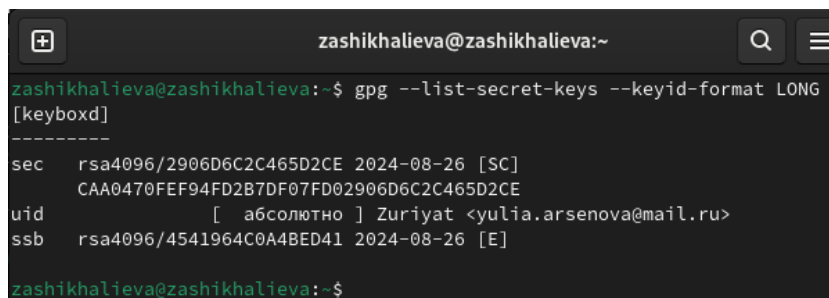


Рис. 10: Аккаунт на Github

Выполнение лабораторной работы. Добавление ключа GPG в Github

- Вывожу список созданных ключей в терминал
- Ищу в результате запроса отпечаток ключа
- Копирую его в буфер обмена (рис. 11).

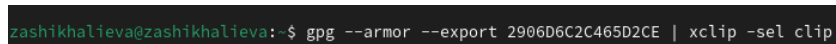


```
zashikhalieva@zashikhalieva:~  
zashikhalieva@zashikhalieva:~$ gpg --list-secret-keys --keyid-format LONG  
[keyboxd]  
-----  
sec   rsa4096/2906D6C2C465D2CE 2024-08-26 [SC]  
      CAA0470FEF94FD2B7DF07FD02906D6C2C465D2CE  
uid   [ абсолютно ] Zuriyat <yulia.arsenova@mail.ru>  
ssb   rsa4096/4541964C0A4BED41 2024-08-26 [E]  
zashikhalieva@zashikhalieva:~$
```

Рис. 11: Вывод списка ключей

Выполнение лабораторной работы. Добавление ключа GPG в Github

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена (рис. 12).



```
zashikhalieva@zashikhalieva:~$ gpg --armor --export 2906D6C2C465D2CE | xclip -sel clip
```

Рис. 12: Копирование ключа в буфер обмена

Выполнение лабораторной работы. Добавление ключа GPG в Github

- Открываю настройки Github, ищу среди них добавление GPG ключа.

- Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. 13).
- Я добавила ключ GPG на GitHub.

Add new GPG key

Title

Key

Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGbMQyABEADLFvbex09NVLbSij7ovgZUKsfr8qCTztu+9sf29fxYcQyomCJO
epzADLIMuymXhrmthWUCcn4Dk8wpwoAveK6O29kun3znNc0YAb1Evsg6BuLIu2UR
yRBpNvlCZF+sOu31RcpWCj1qGp31ShmCuA1uBxz4JEyM1Y68ntyo58+bMURL1Qof
vDtq6Wj82djQ0FvZ+hhSvWkiYboZGjrMlneQ0SfEs9e0nvFkFyt8aRnffRdBs0
pCQNLmijc3Yvtj9/FgbnP4jRiv0lY2mz3OAJ1yGfglMWmkv8mySdYNRoT9e9oxMO
6sMq8Qr6/AMG/B6VGJ4AheeBjMu65j2bX+EwMKMhXQVNHbzO4XplRqYGuInDbEil
5jksWO/tkaMQ7NCSUHGs6rjxtkvNc9QKIO8M2EKPS0Fjii/BI//r+DeRLUzIIIn
```

Add GPG key

Рис. 13: Добавление нового PGP ключа

Выполнение лабораторной работы. Настроить подписи Git

Настраиваю автоматические подписи коммитов git (рис. 14).

```
zashikhalieva@zashikhalieva:~$ git config --global user.signingkey 2906D6C2C465D2CE
zashikhalieva@zashikhalieva:~$ git config --global commit.gpgsign true
zashikhalieva@zashikhalieva:~$ git config --global gpg.program $(which gpg2)
```

Рис. 14: Настройка подписей Git

Выполнение лабораторной работы. Настройка gh

- Начинаю авторизацию в gh
- отвечаю на наводящие вопросы от утилиты
- выбираю авторизоваться через браузер (рис. 15).

```
zashikhalieva@zashikhalieva:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? 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: 04E8-922B
Press Enter to open github.com in your browser...
```

Рис. 15: Авторизация в gh

Выполнение лабораторной работы. Настройка gh

Видю сообщение о завершении авторизации под именем zashikhalieva (рис. 16).

```
! First copy your one-time code: 04E8-922B
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as zashikhalieva
zashikhalieva@zashikhalieva:~$
```

Рис. 16: Завершение авторизации

Выполнение лабораторной работы. Создание репозитория курса на основе шаблона

- Создаю директорию с помощью утилиты mkdir
- Перехожу в только что созданную директорию “Операционные системы”.
- В терминале ввожу команду `gh repo create study_2022-2023_os-intro --template yamadharma/course-directory-student-trmplate --public`.
- После этого клонирую репозиторий к себе в директорию (рис. 17).

```

zashikhalieva@zashikhalieva: /work/study/2022-2023/Операционные системы$ git clone --recursive https://github.com/zashikhalieva/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (32/32), 18.59 КиБ | 288.00 КиБ/с, готово.
Определение изменений: 100% (1/1), 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/zashikhalieva/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0 (from 0)
Получение объектов: 100% (95/95), 96.99 КиБ | 517.00 КиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/zashikhalieva/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0 (from 0)
Получение объектов: 100% (126/126), 335.80 КиБ | 1.04 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8c0b2d67caeb8a19ef8028ced88e'

```

Рис. 17: Создание репозитория

Выполнение лабораторной работы. Создание репозитория курса на основе шаблона

- Перехожу в каталог курса
- Удаляю лишние файлы
- Создаю необходимые каталоги, используя makefile (рис. 18).

```

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

```

Рис. 18: Удаление файлов и создание каталогов

Выполнение лабораторной работы. Создание репозитория курса на основе шаблона

- Сохраняю добавленные изменения
- Комментирую их с помощью git commit (рис. 19).

```
zashikhaliyev@zashikhaliyev: /work/study/2024-2025/Операционные системы/os-intro$ git add .
zashikhaliyev@zashikhaliyev: /work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
```

Рис. 19: Отправка файлов на сервер

Выполнение лабораторной работы. Создание репозитория курса на основе шаблона

Отправляю файлы на сервер с помощью git push (рис. 20).

```
zashikhaliyev@zashikhaliyev: /work/study/2024-2025/Операционные системы/os-intro$ git add .
zashikhaliyev@zashikhaliyev: /work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
```

Рис. 20: Отправка файлов на сервер

Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

Контрольные вопросы

1. Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

Функции системны контроля версий:

- Ведение полной истории всех изменений проекта;
- Возможность возврата назад, с сохранением истории текущих изменений;
- Предоставление информации о том, кто, когда и какие изменения вносил;
- Возможность параллельной работы нескольких людей над одним проектом;

- Возможность разделения проекта на несколько независимых версий.

2. Хранилище - это место, где хранятся все версии файлов проекта.

Commit - операция, при которой изменения, внесенные в рабочую копию, сохраняются в хранилище.

Рабочая копия - это локальная копия проекта, с которой работает разработчик.

Отношения: рабочая копия отправляется в хранилище путем создания коммита. Хранилище содержит историю коммитов.

3. Централизованные системы контроля версий - тип системы контроля версий представляющий собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. Пример: Subversion, CVS

Распределенная система контроля версий - тип системы контроля версий, в которой каждый разработчик имеет полную копию хранилища. Пример: Git, Bazaar.

Отличия: в распределенной системе контроля версий есть локальная копия хранилища у каждого разработчика, что позволяет автономно работать с кодом и нет единого места хранения. В то время как для централизованной системы есть единое место хранения репозитория - сервер. И требует постоянного подключения к интернету при работе.

4. Действия 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"
```

- Настроим utf-8 в выводе сообщений git:

```
git config --global quotepath false
```

- Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

```
mkdir tutorial
```

```
cd tutorial
```

```
git init
```

- После это в каталоге tutorial 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. Порядок работы с общим хранилищем VCS:

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

```
git checkout master
```

```
git pull
```

```
git checkout -b имя_ветки
```

Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральной репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

```
git status
```

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов:

```
git diff
```

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

```
git add ...
```

```
git rm ...
```

Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add
```

Затем сохраняем изменения, поясняя, что было сделано:

```
git commit -am "Some commit message"
```

Отправляем изменения в центральный репозиторий:

```
git push origin имя_ветки
```

или

```
git push
```

6. Основные задачи, решаемые инструментальным средством git:

- Управление версиями
- Совместная одновременная работа над одним проектом
- Хранение истории изменений
- Осуществление контроля доступа
- Резервное копирование

- Создание новых веток

7. Назовите и дайте краткую характеристику командам 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 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 с текущим 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. Примеры использования при работе с локальными репозиториями:

- `git init` создание репозитория
- `git add file.txt`
- `git commit -am "New file"`
- `git status`

Примеры использования при работе с удаленными репозиториями:

- `git remote add origin`
 - `ssh://git@github.com//.git`
 - `git push -u origin master`
9. Ветви - это набор коммитов, расположенных в хронологическом порядке. Ветви нужны, чтобы разработчики вели совместно работу над проектом и не мешали другим и не сломать основную программу, в случае ошибки.
10. Игнорировать файлы можно с помощью файла `.gitignore`, например, при работе создаются файлы как и системой, так и объектными редакторами, которые не требуются добавлять в репозиторий, что позволяет не загружать их. А делается это следующим образом: 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
```