

Северо-Кавказский федеральный университет

Кафедра инфокоммуникаций СКФУ

Отчет
По лабораторной работе №1
По предмету: «Основы
кроссплатформенного программирования»

Исполнитель:

Студента группы ИТС-б-з-22-1

Направление подготовки 11.03.02
Инфокоммуникационные
технологии и системы связи

Пальников Станислав Петрович

(Ф.И.О.)

Руководитель дисциплины:

Воронкин Роман Александрович

Ставрополь, 2023

Ответы на контрольные вопросы:

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Локальные – частый локальный запуск и отладка при внесении небольших изменений, необходимость взаимодействовать с другими разработчиками

Централизованные - единая точка отказа, если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

3. К какой СКВ относится Git?

Git относится к распределённым системам контроля версий

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего

проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль - это ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8. Какие бывают репозитории в GitHub?

Каждый проект размещается в своем собственном контейнере, который называется репозиторием. В нем можно хранить код, конфигурации, наборы данных, изображения и другие файлы, включенные в ваш проект. Любые изменения файлов в репозитории будут отслеживаться с помощью контроля версий. Во вкладке Code находятся два файла. README.md - это файл, который описывает проект; каждый репозиторий должен включать этот файл. GitHub находит его и отображает его содержимое под репозиторием. Другой файл – .gitignore – указывает, какие файлы и каталоги Git следует игнорировать.

9. Укажите основные этапы модели работы с GitHub.

Стандартный подход к работе с проектом состоит в том, чтобы иметь локальную копию репозитория и фиксировать ваши изменения в этой копии, а не в удаленном репозитории, размещенном на GitHub. Этот локальный репозиторий имеет полную историю версий проекта, которая может быть полезна при разработке без подключения к интернету. После того, как вы что-то изменили в локальном, вы можете отправить свои изменения в удаленный репозиторий, чтобы сделать их видимыми для других разработчиков. Если вы хотите внести небольшие изменения в свою копию (fork), вы можете использовать вебинтерфейс GitHub. Однако такой подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, введите команду ниже в терминале, чтобы отобразить текущую версию вашего Git:

```
git version
```

Если она сработала, давайте добавим в настройки Git ваше имя, фамилию и адрес электронной почты, связанный с вашей учетной записью GitHub:

```
git config --global user.name
```

```
git config --global user.email <EMAIL>
```

11. Опишите этапы создания репозитория в GitHub.

Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.

Описание (Description). Можно оставить пустым.

Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Свободного использования: открытая, бесплатная, условно-бесплатная.

Несвободного использования: коммерческая, условно-бесплатная.

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

После создания репозитория его необходимо клонировать на компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите git clone и введите адрес.

14. Как проверить состояние локального репозитория Git?

Чтобы проверить состояние локального репозитория нужно ввести команду:
git status.

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/ измененного файла под версионный контроль с помощью команды `git add` ; фиксации (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push` ?

После изменения файла: «modified: README.md»

После добавления файла: «git add README.md»

После фиксации изменений: «git commit -m "Add information about local repository in readme file"»

После отправки на сервер: «git push --set-upstream origin edit-readme»

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды `git clone` .

`git clone` для копирования репозитория

периодически выполнять `git pull`, особенно перед `git push`, чтобы предотвратить возможные конфликты

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

Популярные сервисы с Git: BitBucket, GitLab, Google Code. Если сравнивать GitHub и GitLab: GitHub делает упор на высокую доступность и производительность своей инфраструктуры и делегирует другие сложные функции сторонним инструментам. GitLab, наоборот, фокусируется на

включении всех функций на одной проверенной и хорошо интегрированной платформе; он обеспечивает все для полного жизненного цикла DevOps под одной крышей. Что касается популярности, GitHub определенно превосходит GitLab. В GitLab меньше разработчиков внедряют на платформу открытые исходные коды. Кроме того, что касается цен, GitHub стоит дороже, что делает его неподходящим для пользователей с небольшим бюджетом.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств

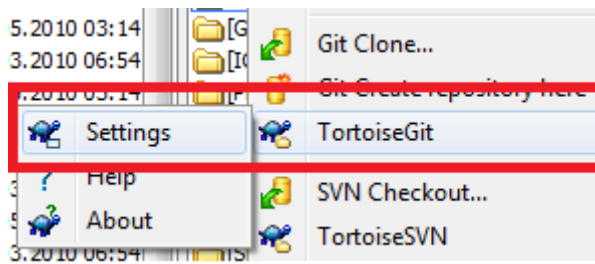
TortoiseGit — это бесплатный клиентский инструмент с открытым исходным кодом для репозитория на основе Git, который управляет пользовательскими файлами и отслеживает их изменения. Реализован как расширение проводника Windows (shell extension). Подписывает иконки к файлам, находящимся под управлением Git, для отображения их статуса в Git.

TortoiseGit поддерживает выполнение обычных задач:

- создание коммитов;
- отображение журналов;
- сравнение двух версий;
- создание веток и тегов;
- создание исправлений и т. д.

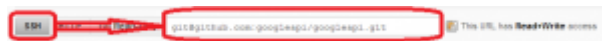
Клонирование с помощью TortoiseGit:

1) Вызываем контекстное меню и выбираем пункт «*TortoiseGit — Settings*»:

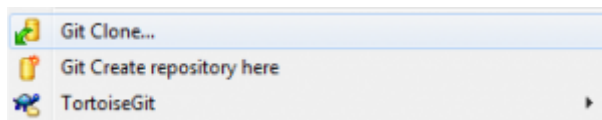


В появившемся окне переходим сразу к пункту «**Git — Config**» и записываем свое имя и адрес электронной почты. Эти данные **должны в точности совпадать** с теми, что записаны в Вашем аккаунте на github, иначе ваш ключ просто не подойдет.

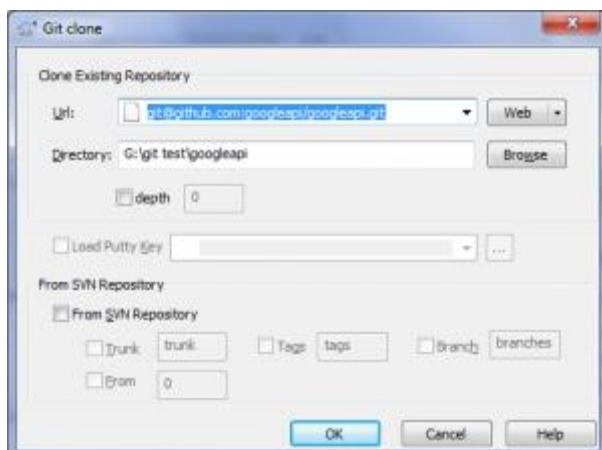
2) Клонирование репозитория. Для этого заходим на страницу проекта, и копируем в буфер адрес:



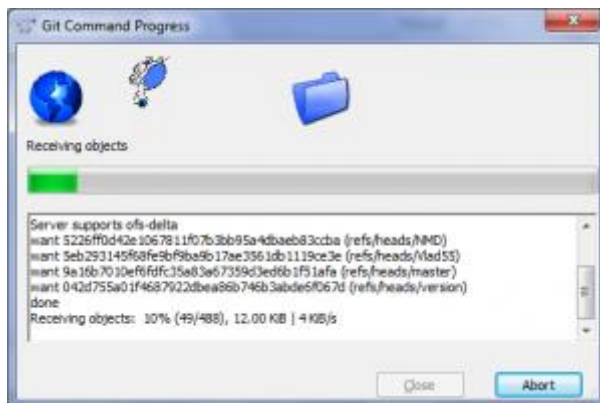
Теперь жмем правой кнопкой мыши на директории, в которой будем хранить исходники и в меню выбираем «Git Clone..»:



В открывшемся окне в поле URL вставляем скопированный адрес и жмем «Ok»:



Начнется процесс клонирования репозитория.



Всё вышесказанное можно было бы заменить всего двумя командами в консоли:

```
cd path/to/dir  
git clone URL
```

После клонирования репозитория Вы автоматически переключитесь на нашу главную ветку (master). Так как каждый из нас занят определенной работой в проекте, то у каждого своя ветвь в репозитории, поэтому и Вам придется создавать свой branch. Делается это достаточно просто.

Выполнение работы:

2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после

установки флажка Add .gitignore).

Search or jump to...

Pull requests

Issues

Codespaces

Marketplace

Explore

+

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

Repository name *

free-proger

1.1

1.1 is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-octo-telegram](#) ?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

Create repository

3. Выполните клонирование созданного репозитория на рабочий компьютер.

Search or jump to...

Pull requests

Issues

Codespaces

Marketplace

Explore

+

free-proger / 1.1

Public

Pin

Unwatch

Fork

Star

< Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

1 branch

0 tags

Go to file

Add file

Code

free-proger Initial commit

LICENSE Initial commit

README.md Initial commit

README.md

1.1

Local

Codespaces

Clone

HTTPS SSH GitHub CLI

https://github.com/free-proger/1.1.git

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

MIT license

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2023 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact GitHub

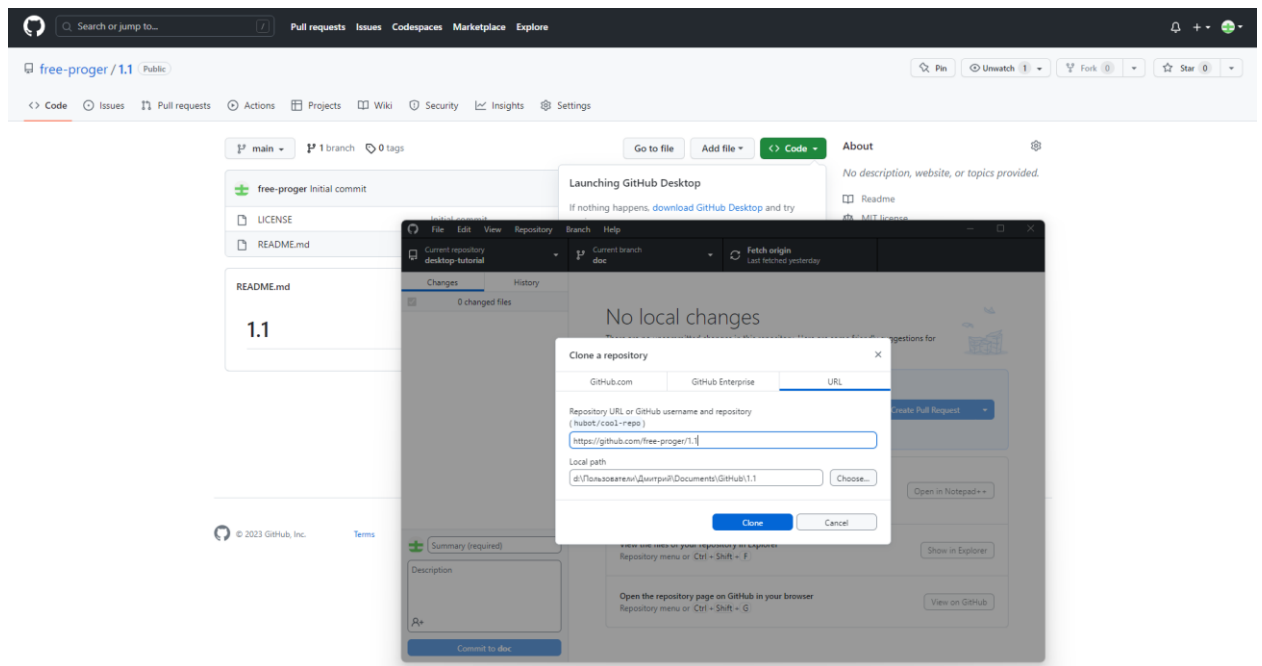
Pricing

API

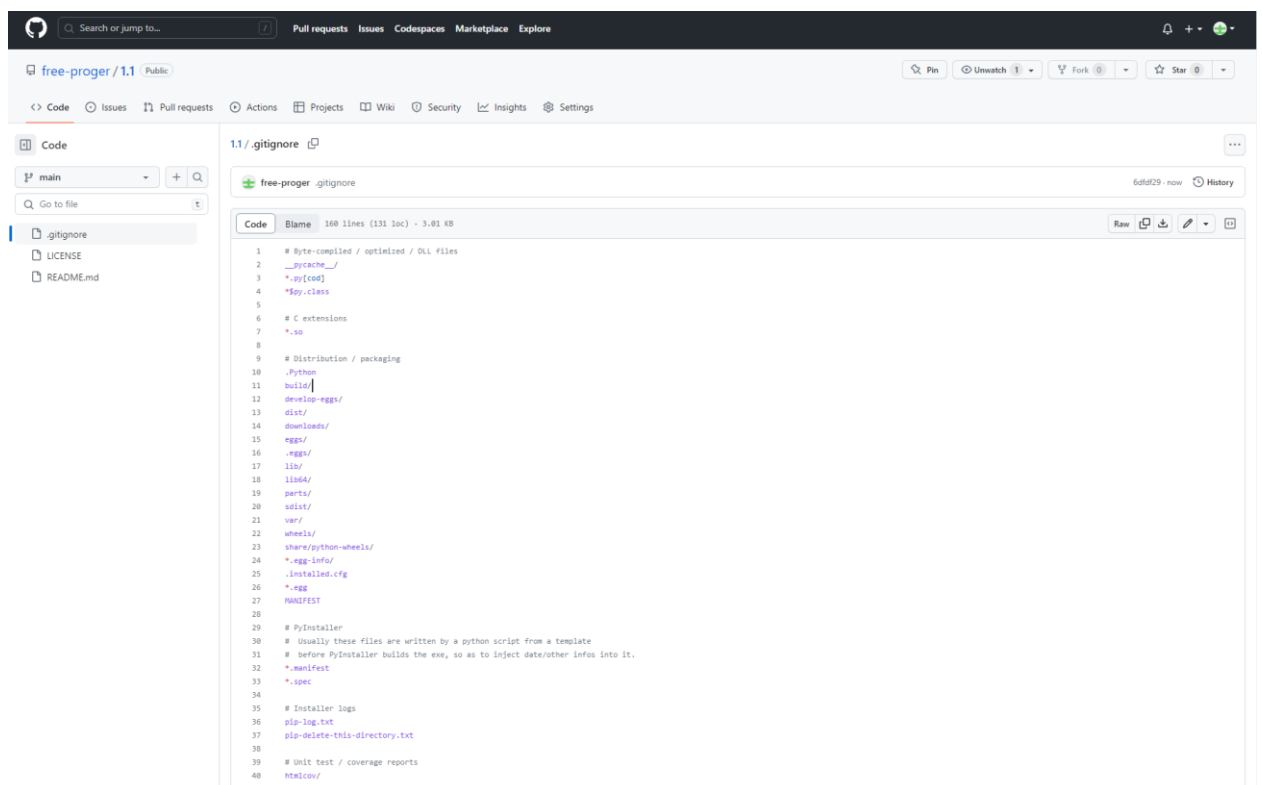
Training

Blog

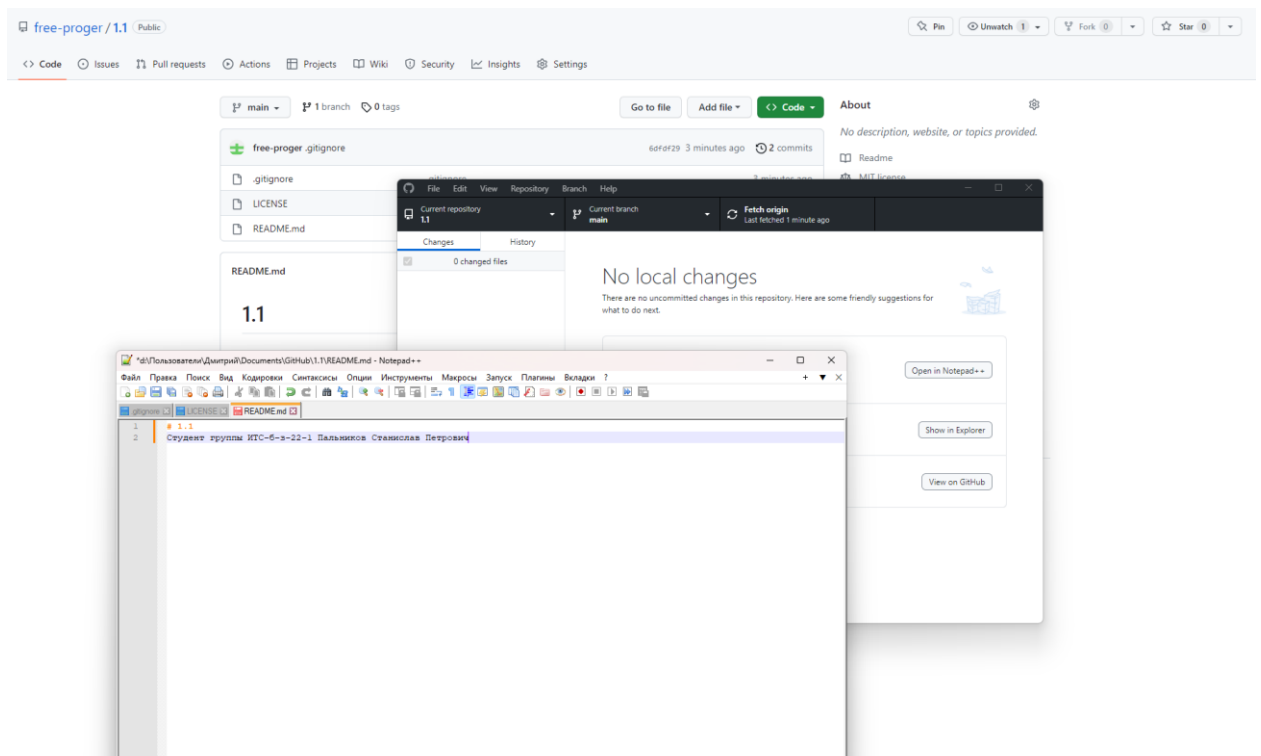
About



4. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



5. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.



6. Напишите небольшую программу на выбранном Вами языке программирования. Фиксируйте изменения при написании программы в локальном репозитории. Должно быть сделано не менее 7 коммитов.

