

Лабораторная работа №2

Первоначальная настройка git.

Хохлачёва Полина Дмитриевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6

Список иллюстраций

2.1	Установка	6
2.2	Задаём имя	6
2.3	Параметры	6
2.4	Создание	7
2.5	Создание	8
2.6	Создание	8
2.7	Настройка	9
2.8	Создание	9
2.9	Настройка	9
2.10	Удаление и создание	10
2.11	Отправление	10

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Установка программного обеспечения(рис. 2.1).

```
khokhlachevapolina@khokhlachevapolina:~$ dnf install git
Для выполнения запрошенной операции требуются привилегии суперпользователя. Пожалуйста, войдите в систему как пользователь с повышенными правами или используйте опции '--assumeno' или '--downloadonly', чтобы выполнить команду без изменения состояния системы.
khokhlachevapolina@khokhlachevapolina:~$ sudo dnf install git
[sudo] пароль для khokhlachevapolina:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет 'git-2.48.1-1.fc41.x86_64' уже установлен.

Нечего делать.
khokhlachevapolina@khokhlachevapolina:~$ dnf install gh
Для выполнения запрошенной операции требуются привилегии суперпользователя. Пожалуйста, войдите в систему как пользователь с повышенными правами или используйте опции '--assumeno' или '--downloadonly', чтобы выполнить команду без изменения состояния системы.
khokhlachevapolina@khokhlachevapolina:~$
```

Рис. 2.1: Установка

Задаём имя и email репозитория(рис. 2.2).

```
khokhlachevapolina@khokhlachevapolina:~$ git config --global user.name 'Name Surname'
khokhlachevapolina@khokhlachevapolina:~$ git config --global user.email 'work@mail'
khokhlachevapolina@khokhlachevapolina:~$
```

Рис. 2.2: Задаём имя

Параметры(рис. 2.3).

```
cr
khokhlachevapolina@khokhlachevapolina:~$ git config --global core.autocrlf input
khokhlachevapolina@khokhlachevapolina:~$ ^C
khokhlachevapolina@khokhlachevapolina:~$ git config --global core.autocrlf input
khokhlachevapolina@khokhlachevapolina:~$
```

Рис. 2.3: Параметры

Создание ключей(рис. 2.4).

```
Enter passphrase for /home/khokhlaacheva/.ssh/id_rsa:
):
Enter same passphrase again:
Your identification has been saved in /home/khokhlaacheva/.ssh/id_rsa
Your public key has been saved in /home/khokhlaacheva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:eOu3Ak+72r0YA2616v1JEhGCSpx1b1oDN7b+JaRry7c khokhlaacheva@lin
The key's randomart image is:
+---[RSA 4096]---+
|o+ . . . . |
|ooB . . . . |
|==+ . . . . |
|+o . . . o . |
|oo . S oo |
|o . . o oo |
| oo o ... + |
|.oo . .o+=... |
|.ooE. .o+=.*+. |
+---[SHA256]---+
[khokhlaacheva@lin ~]$
```

Рис. 2.4: Создание

создание ключей pgr(рис. 2.5).


```

-----END PGP PUBLIC KEY BLOCK-----
[khokhlachevapolina@khokhlachevapolina ~]$ git config --global user.signingkey khokhlachevapolina@gmail.com
[khokhlachevapolina@khokhlachevapolina ~]$ git config --global commit.gpgsign true
[khokhlachevapolina@khokhlachevapolina ~]$ git config --global gpg.program $(which gpg2)
[khokhlachevapolina@khokhlachevapolina ~]$

```

Рис. 2.7: Настройка

создание репозитория на основе шаблона(рис. 2.8).

```

[khokhlachevapolina@khokhlachevapolina ~]$ mkdir -p ~/work/study/2024-2025/Операционные системы
[khokhlachevapolina@khokhlachevapolina ~]$ cd ~/work/study/2024-2025/Операционные системы
[khokhlachevapolina@khokhlachevapolina ~]$ gh repo create study_2022-2023_os-intro --template=yanadharma/course-directory-student-template --public
✓ Created repository companchelo/study_2022-2023_os-intro on GitHub
https://github.com/companchelo/study_2022-2023_os-intro
[khokhlachevapolina@khokhlachevapolina ~]$

```

Рис. 2.8: Создание

настройка каталога курса(рис. 2.9).

```

[khokhlachevapolina@khokhlachevapolina ~]$ cd ~/work/study/2024-2025/Операционные системы/os-intro
[khokhlachevapolina@khokhlachevapolina ~]$

```

Рис. 2.9: Настройка

удаление лишних файлов и создание необходимых каталогов(рис. ??).

```
n 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 810.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5
086c9c72a02bd2fca1d4a6'
[khokhlachevapolina@khokhlachevapolina Операционные системы]$ cd ~/work
/study/2024-2025/'Операционные системы'/os-intro
[khokhlachevapolina@khokhlachevapolina os-intro]$ rm package.json
[khokhlachevapolina@khokhlachevapolina os-intro]$ echo os-intro > COURSE
[khokhlachevapolina@khokhlachevapolina os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules
[khokhlachevapolina@khokhlachevapolina os-intro]$
```

Рис. 2.10: Удаление и создание

отправляем файл(рис. ??).

```
create node 100644 project-personal/stage6/report/report.md
[khokhlachevapolina@khokhlachevapolina os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При скатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.30 КиБ | 1.17 МБ/с, готово.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:comanchelo/study_2024-2025_os-intro.git
 108615e..33963d9 master -> master
[khokhlachevapolina@khokhlachevapolina os-intro]$
```

Рис. 2.11: Отправление

Ответы на вопросы

□1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (Version Control Systems, VCS) — это инструменты, которые позволяют отслеживать изменения в файлах и координировать работу нескольких людей над одним проектом. Основные задачи, которые решают VCS, включают:

- Отслеживание изменений в коде и других файлах.
- Сохранение историй изменений, что позволяет откатываться к предыдущим версиям.
- Совместная работа над проектом несколькими разработчиками.
- Управление конфликтами при одновременной работе нескольких пользователей над одним файлом.

□2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище (repository) — это место, где хранятся все версии проекта, включая файлы и их изменения.
- Commit — это операция, которая сохраняет текущее состояние файлов в хранилище, фиксируя изменения с комментарием, описывающим эти изменения.
- История — это последовательность всех коммитов в репозитории, позволяющая отслеживать, какие изменения были внесены и когда.
- Рабочая копия (working copy) — это локальная версия файлов проекта, с которой разработчик работает на своем компьютере. Она может содержать изменения, которые еще не были зафиксированы (commit).

□3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

- Централизованные VCS: Все изменения хранятся на одном центральном сервере. Пользователи получают доступ к этому серверу для получения последних изменений и отправки своих. Пример: Subversion (SVN), CVS.
- Децентрализованные VCS: Каждый разработчик имеет полную копию репозитория, включая всю историю изменений. Это позволяет работать офлайн и синхронизировать изменения с другими разработчиками по мере необходимости. Пример: Git, Mercurial.

□4. Опишите действия с VCS при единоличной работе с хранилищем.

При единоличной работе с VCS разработчик обычно выполняет следующие действия:

1. Создание репозитория: Инициализация нового репозитория с помощью команды `git init`.
2. Работа с файлами: Создание или редактирование файлов в рабочей копии.
3. Добавление изменений: Использование команды `git add` для добавления измененных файлов в индекс.
4. Коммит изменений: Фиксация изменений с помощью команды `git commit`.
5. Просмотр истории: Использование команды `git log` для просмотра истории коммитов.
6. Откат изменений: При необходимости можно откатить изменения к предыдущему коммиту.

□5. Опишите порядок работы с общим хранилищем VCS.

При работе с общим хранилищем VCS порядок действий может быть следующим:

1. Клонирование репозитория: Получение копии удаленного репозитория с помощью `git clone`.
2. Создание новой ветки: Создание новой ветки для работы над новой функцией или исправлением.
3. Внесение изменений: Работа с файлами в рабочей копии.
4. Добавление и коммит изменений: Использование `git add` и `git commit` для сохранения изменений.
5. Синхронизация с удаленным репозиторием: Использование `git pull` для получения последних изменений из удаленного репозитория.
6. Отправка изменений: Использование `git push` для отправки своих коммитов в удаленный репозиторий.

□6. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи Git включают:

- Отслеживание изменений в коде.
- Ведение истории всех изменений.
- Управление ветвями и слияниями.
- Поддержка совместной работы нескольких разработчиков.
- Упрощение отката изменений и исправления ошибок.

□7. Назовите и дайте краткую характеристику командам git.

- git init: Инициализация нового репозитория.
- git clone: Клонирование удаленного репозитория.
- git add: Добавление изменений в индекс для последующего коммита.
- git commit: Фиксация изменений в локальном репозитории.
- git status: Просмотр статуса рабочей копии (изменения, которые еще не закоммичены).
- git log: Просмотр истории коммитов. # Выводы

Мы изучили идеологию и применение средств контроля версий. Освоили умения по работе с git.