

Лабораторная работа номер 2

Выполнение лабораторной работы

Титков Ярослав Максимович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	12
6	Контрольные вопросы:	13

Список иллюстраций

4.1	Установка Git	9
4.2	Создание ключа	9
4.3	Настройка конфигов	10
4.4	Получение индивидуального кода	10
4.5	Клонирование репозитория	10
4.6	Создание ветки	11

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с `git`.

2 Задание

1.Создание Гит ключей 2.Подключение репозитория 3. Работа с GitHub

3 Теоретическое введение

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 дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными

участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

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.

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 Выполнение лабораторной работы

```
root@yitkov:~# dnf install git
Для выполнения запрошенной операции требуется привилегии суперпользователя. Пожалуйста, войдите в систему как пользователь с повышенными правами или используйте опции "--assumeno" или "--downloadonly", чтобы выполнить команду без изменения состояния системы.
root@yitkov:~# sudo -i
[sudo] пароль для yitkov:
root@yitkov:~# dnf install git
-bash: dnf: команда не найдена
root@yitkov:~# dnf install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - updates
Fedora 41 - x86_64 - updates
Репозитории загрузки:
Пакет "git-2.46.1-1.fc41.x86_64" уже установлен.

Нечего делать.
root@yitkov:~# dnf install git
Обновление и загрузка репозитория:
Репозитории загрузки:
Пакет
Установка:
git
Арк. Версия Репозиторий Размер
x86_64 2.46.0-1.fc41 updates 42.6 MiB
Сводка транзакции:
Установка: 1 пакета
Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
Is this ok [y/N]: y
(1/1) git-2.46.0-1.fc41.x86_64
-----
(1/1) Total:
-----
100% | 2.1 MiB/s | 10.3 MiB | 00m05s
100% | 1.6 MiB/s | 10.3 MiB | 00m07s
(1/1) Проверить файлы пакета
(2/3) Подготовить транзакцию
(3/3) Установка git-2.46.0-1.fc41.x86_64
Завершено!
root@yitkov:~# git config --global core.quietfetch false
root@yitkov:~# git config --global init.defaultBranch master
root@yitkov:~#
```

Рис. 4.1: Установка Git

```
0. +00.0. |
|+. .0 +0 |
|. .+00. |
| .0=0 |
+----[SHA256]-----+
root@yitkov:~# gpg --full-generate-key
gpg (GnuPG) 2.4.5; 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 '/root/.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

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Yarick
Адрес электронной почты: Mandusrek@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
```

Рис. 4.2: Создание ключа

```
root@yitikov:~# gpg --list-secret-keys --keyid-format LONG
-----
sec   rsa4096/57CE861E85A21AD 2025-03-05 [SC]
      5D61CBB83AC799FC11B4D57CE861E85A21AD
uid   [ unknown ] Varic <VaricVaros@gmail.com>
ssb   rsa4096/962DAE1BA5F4C92 2025-03-05 [E]

root@yitikov:~# gpg --armor --export 57CE861E85A21AD | xclip -sel clip
bash: xclip: команда не найдена
gpg: [stdin]: write error: Ошибка канала
gpg: filter: flush failed on close: Ошибка канала
root@yitikov:~# dnf install xclip
Обновление и загрузка репозитория:
Репозитории загрузки:

Пакет                                     Арх.          Версия                Репозиторий          Размер
-----
Установка:
xclip                                     x86_64        0.13-22.git11cba61.fc41  fedora                62.4 KiB

Сводка транзакции:
Установка:      1 пакета

Общий размер входящих пакетов составляет 37 KiB. Необходимо загрузить 37 KiB.
После этой операции будут использоваться дополнительные 62 KiB (установка 62 KiB, удаление 0 B).
Is this ok [y/N]: y
(1/1) xclip-0:0.13-22.git11cba61.fc41.x86_64
-----
100% | 589.3 KiB/s | 36.5 KiB | 00m00s
(1/1) Total
-----
100% | 131.4 KiB/s | 36.5 KiB | 00m00s
Выполнение транзакции
(1/2) Проверить файлы пакета
(2/3) Подготовить транзакцию
(3/3) Установка xclip-0:0.13-22.git11cba61.fc41.x86_64
завершено!
root@yitikov:~# gpg --armor --export 57CE861E85A21AD | xclip -sel clip
root@yitikov:~# gpg --armor --export 57CE861E85A21AD | xclip -sel clip
root@yitikov:~# git config --global user.signingkey 57CE861E85A21AD
root@yitikov:~# git config --global commit.gpgsign true
root@yitikov:~# git config --global gpg.program S(which gpg2)
root@yitikov:~#
```

Рис. 4.3: Настройка конфигов

```
yitikov@yitikov:~$ gh auth login
? Where do you use GitHub? 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: 0136-ED96
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as TitkovVaroslav
yitikov@yitikov:~$
```

Рис. 4.4: Получение индивидуального кода

```
bash: owner: Нет такого файла или каталога
yitikov@yitikov:~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com:owner/study_2024-2025-os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.37 KiB | 19.37 MiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подготовка <template/presentation> (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован на пути <template/presentation>
Подготовка <template/report> (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован на пути <template/report>
Клонирование в <home/yitikov/work/study/2024-2025/Операционные системы/os-intro/template/presentation>...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 142.17 KiB | 86.00 KiB/c, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в <home/yitikov/work/study/2024-2025/Операционные системы/os-intro/template/report>...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 KiB | 821.00 KiB/c, готово.
Определение изменений: 100% (48/48), готово.
Submodule path 'template/presentation': checked out 'c9e2712b4b2d431ad5086c9c72a82b2fca1da6'
Submodule path 'template/report': checked out 'c26e22effe78e6d95f0d2ef55b1a1b5f5c769'
yitikov@yitikov:~/work/study/2024-2025/Операционные системы$ cd -> /work/study/2024-2025/Операционные системы/os-intro
yitikov@yitikov:~/work/study/2024-2025/Операционные системы/os-intro$
```

Рис. 4.5: Клонирование репозитория

```

titkov@ytitkov: ~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
titkov@ytitkov:~/work/study/2024-2025/Операционные системы/os-intro$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules

titkov@ytitkov:~/work/study/2024-2025/Операционные системы/os-intro$ git add .
titkov@ytitkov:~/work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master 8f75e2f] feat(main): make course structure
 1 file changed, 1 insertion(+), 14 deletions(-)
Committer: Varoslav Titkov <ytitkov@ytitkov.net>
Ваше имя или электронная почта настроены автоматически на основании вашего имени пользователя и имени машины. Пожалуйста, проверьте, что они оправданы правильно.
Вы можете отключить это уведомление установив их напрямую:

  git config --global user.name "Ваше Имя"
  git config --global user.email you@example.com

После этого, изменить авторство этой коммита можно будет с помощью команды:

  git commit --amend --reset-author

2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
titkov@ytitkov:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 295 байтов | 295.00 Кб/с, готово.
Total: 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:TitkovVaroslav/study_2024-2025_os-intro.git
   bb2b34b..8f75e2f  master -> master
titkov@ytitkov:~/work/study/2024-2025/Операционные системы/os-intro$

```

Рис. 4.6: Создание ветки

5 Выводы

В ходе работы я освоил и научился пользоваться Git

6 Контрольные вопросы:

Системы контроля версий (VCS)

Инструменты для управления изменениями в файлах. Используются для истории изменений, совместной работы и отката. Основные понятия VCS:

Хранилище (репозиторий) – база данных с версиями файлов.

Commit – фиксация изменений с комментарием.

История – последовательность коммитов.

Рабочая копия – локальные файлы для работы.

Централизованные и децентрализованные VCS

Централизованные (CVCS): Одно главное хранилище. Пример: SVN.

Децентрализованные (DVCS): У каждого своя копия репозитория. Пример: Git, Mercurial.

Действия с VCS при единоличной работе:

Создать репозиторий (`git init`).

Добавить файлы (`git add`).

Фиксировать изменения (`git commit`).

Просматривать историю (`git log`).

Работа с общим хранилищем:

Клонировать репозиторий (`git clone`).

Получать изменения (`git pull`).

Отправлять изменения (`git push`).

Решать конфликты при необходимости.

Основные задачи Git:

Управление версиями.

Совместная работа.

Ветвление и слияние.

Отслеживание изменений.

Основные команды Git:

`git init` – создать репозиторий.

`git add` – добавить файлы в индекс.

`git commit` – зафиксировать изменения.

`git status` – проверить состояние.

`git log` – посмотреть историю.

`git clone` – скопировать репозиторий.

`git pull` – получить изменения.

`git push` – отправить изменения.

`git branch` – управление ветками.

`git merge` – слияние веток.

Примеры работы с Git:

Локальный репозиторий:

```
git init
git add file.txt
git commit -m "Initial commit"
```

Удаленный репозиторий:

```
git clone https://github.com/user/repo.git
git pull
git push origin main
```

Ветви (branches)

Используются для изоляции изменений (например, для новой функции или исправления).

Создать ветку: `git branch new-feature`.

Переключиться на ветку: `git checkout new-feature`.

Игнорирование файлов

Файл `.gitignore` указывает, какие файлы не добавлять в репозиторий (например, временные файлы).

Пример `.gitignore`: