

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

*дисциплина: Операционные системы*

Студент: Желдакова Виктория Алексеевна

Студ. Билет №1032216445

Группа: НФИбд-01-21

**МОСКВА**

2022 г.

## Управление версиями

### Цель работы:

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

### Теоретическое введение:

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.

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.

## Выполнение работы:

Во-первых, мы создали учётную запись на github.com и заполнили основные данные о себе.

Затем установили git-flow (Рис.1.1, Рис.1.2) и gh (Рис.2).

```
[vazheldakova@fedora ~]$ cd /tmp
[vazheldakova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[vazheldakova@fedora tmp]$ ls
gitflow-installer.sh
[vazheldakova@fedora tmp]$ chmod +x gitflow-installer.sh
[vazheldakova@fedora tmp]$ ls
gitflow-installer.sh
[vazheldakova@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 vazheldakova:
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 МБ | 1.90 МБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
Install: создание каталога '/usr/local/share/doc/'
Install: создание каталога '/usr/local/share/doc/gitflow/'
Install: создание каталога '/usr/local/share/doc/gitflow/hooks/'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
'gitflow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
'gitflow/hooks/filter-flow-release-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-finish-tag-message'
'gitflow/hooks/filter-flow-release-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-start-version'
```

Рис.1.1 Скачивание установщика git-flow и изменение его прав доступа

```
[vazheldakova@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 vazheldakova:
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 МБ | 1.90 МБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
Install: создание каталога '/usr/local/share/doc/'
Install: создание каталога '/usr/local/share/doc/gitflow/'
Install: создание каталога '/usr/local/share/doc/gitflow/hooks/'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
'gitflow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
'gitflow/hooks/filter-flow-release-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-finish-tag-message'
'gitflow/hooks/filter-flow-release-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-start-version'
```

Рис.1.2 Установка git-flow

```
[vazheldakova@fedora tmp]$ sudo dnf install gh
Fedora 35 - x86_64 - Updates                                19 kB/s | 19 kB  00:00
Fedora Modular 35 - x86_64 - Updates                       4.2 kB/s | 18 kB  00:04
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                                806 kB/s | 6.8 MB  00:08
=====
Общий размер
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка:
Установка      : gh-2.7.0-1.fc35.x86_64
Запуск скрипта : gh-2.7.0-1.fc35.x86_64
Проверка       : gh-2.7.0-1.fc35.x86_64
=====
Установлен:
gh-2.7.0-1.fc35.x86_64
=====
Выполнено!
```

Рис.2 Установка gh

Затем задали имя и email владельца репозитория, настроили utf-8 в выводе сообщений git, а также верификацию и подписание коммитов git. Дали имя «master» начальной ветке. Настроили параметры autocrlf и safecrlf.

```
[vazheldakova@fedora tmp]$ git config --global user.name Viktoria Zheldakova
[vazheldakova@fedora tmp]$ git config --global user.email v.zhel3ynadex.ru
[vazheldakova@fedora tmp]$ git config --global core.quotepath false
[vazheldakova@fedora tmp]$ git config --global init.defaultBranch master
[vazheldakova@fedora tmp]$ git config --global core.autocrlf input
[vazheldakova@fedora tmp]$ git config --global core.safecrlf warn
[vazheldakova@fedora tmp]$ git config --global core.autocrlf input
```

Рис.3 Базовая настройка git

Создали ключи SSH по алгоритмам RSA и ed25519.

```
[vazheldakova@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vazheldakova/.ssh/id_rsa):
Created directory '/home/vazheldakova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vazheldakova/.ssh/id_rsa
Your public key has been saved in /home/vazheldakova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:SUPm0kye5JAXz1MeSoj9m2Y7gbH1hN9V52cYrnbzjg vazheldakova@fedora
The key's randomart image is:
+----[RSA 4096]-----+
|  .+*0. 0 0 + |
| 0%+= + = +0 |
| ..X.= 0 0.. |
| 0.00. .. + |
| $= + .0.0 |
| 0 * 0.0. |
| 0 + +. |
| 0 E 0 |
| . . |
+----[SHA256]-----+
```

Рис.4.1 Создание SSH ключа по алгоритму RSA

```
[vazheldakova@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/vazheldakova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vazheldakova/.ssh/id_ed25519
Your public key has been saved in /home/vazheldakova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:z8MpIUyb51fm+btiqueDcy6Q+Vxc09fHWPhyMjtGLRmU vazheldakova@fedora
The key's randomart image is:
---[ED25519 256]---+
|
|      E.. |
|    o .  +.o.+ |
|  . =  +.+..o= |
| + $ o Boo+. |
| . . +.=.+oo. |
| ooB. o. . |
| oo+++. |
| .++o+ .o. |
|-----[SHA256]-----+
```

Рис.4.2 Создание SSH ключа по алгоритму ed25519

Сгенерировали GPG ключ.

```
[vazheldakova@fedora tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; Copyright (C) 2021 Free Software Foundation, Inc.
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/vazheldakova/.gnupg'
gpg: создан шит с ключами '/home/vazheldakova/.gnupg/pubring.kbx'
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: Viktoria
Адрес электронной почты: v.zhel3@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"Viktoria <v.zhel3@yandex.ru>"
```

Рис.5.1 Настройка генерации GPG ключа

```
gpg: /home/vazheldakova/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ C798C5DD4883E8C7 помечен как абсолютно доверенный
gpg: создан каталог '/home/vazheldakova/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/vazheldakova/.gnupg/openpgp-revocs.d/6173DAB8D6F7C4837D378B2EC798C5DD4883E8C7.rev'
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-23 [SC]
      6173DAB8D6F7C4837D378B2EC798C5DD4883E8C7
uid           Viktoria <v.zhel3@yandex.ru>
sub   rsa4096 2022-04-23 [E]
```

Рис.5.2. Результат генерации GPG ключа



Вывели список ключей и скопировали отпечаток приватного ключа, затем скопировали сгенерированный ключ в буфер обмена и вставили полученный ключ в поле ввода в настройках GitHub. (Так же пришлось установить утилиту xclip для копирования в буфер).

```
[vazheldakova@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/vazheldakova/.gnupg/pubring.kbx
-----
sec   rsa4096/C790C5DD4883E8C7 2022-04-23 [SC]
      6173DAB8D6F7C4837D378B2EC790C5DD4883E8C7
uid           [ абсолютно ] Viktoria <v.zhel3@yandex.ru>
ssb   rsa4096/F32EC9BF6E3A84CC 2022-04-23 [E]

[vazheldakova@fedora tmp]$ gpg --armor --export C790C5DD4883E8C7 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

+ Waiting in queue...
The following packages have to be installed:
xclip-0.13-15.git11cba61.fc35.x86_64  Command line clipboard grabber
Proceed with changes? [N/y] y

+ Waiting in queue...
+ Waiting for authentication...
+ Waiting in queue...
+ Downloading packages...
+ Requesting data...
+ Testing changes...
+ Installing packages...
```

Рис.6.1 Вывод списка ключей и установка xclip

```
[vazheldakova@fedora tmp]$ gpg --armor --export C790C5DD4883E8C7 | xclip -sel clip
```

Рис.6.2 Добавление ключа в буфер обмена

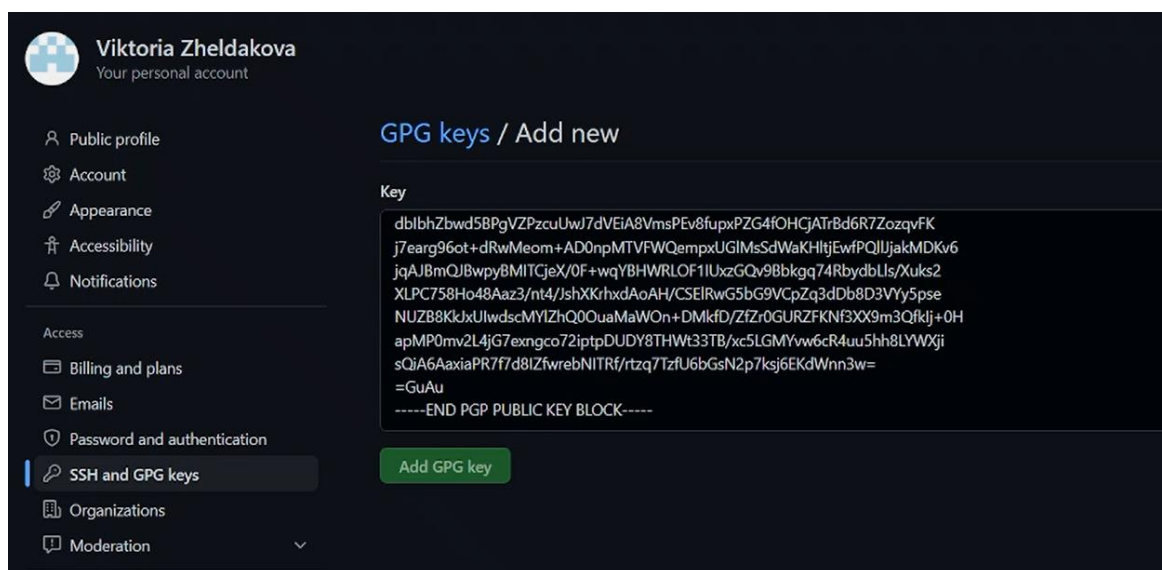


Рис.6.3 Добавление ключа в настройки GitHub

Настроили автоматическую подпись коммитов git.

```
[vazheldakova@fedora tmp]$ git config --global user.signingkey C790C5DD4883E8C7
[vazheldakova@fedora tmp]$ git config --global commit.gpgsign true
[vazheldakova@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

Рис. 7 Настройка подписи коммитов

Затем авторизовались на github.com

```
[vazheldakova@fedora tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? 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: 936D-7547
Press Enter to open github.com in your browser...
```

Рис.8 Авторизация

Создали каталог «Операционные системы», в нём создали репозиторий и клонировали шаблон.

```
[vazheldakova@fedora ~]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[vazheldakova@fedora ~]$ cd ~/work/study/2021-2022/"Операционные системы"
[vazheldakova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository bermuly/study_2021-2022_os-intro on GitHub
```

Рис.9.1 Создание каталога и репозитория в нём

```
[vazheldakova@fedora Операционные системы]$ git clone --recursive git@github.com:bermuly/study_2021-2022_os-intro.git
os-intro
Клонирование в «os-intro»...
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), 12.50 КиБ | 12.50 КиБ/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/vazheldakova/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 9), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 КиБ | 742.00 КиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/home/vazheldakova/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 | 1.79 КиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eae6b7586f8a9d...»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef88f8def3b9a496...»
```

Рис.9.2 Клонирование шаблона

Перешли в каталог курса, удалили лишние файлы, создали необходимые каталоги и отправили файлы на сервер.



```
[vazheldakova@fedora Операционные системы]$ cd ~/work/study/2021-2022/"O nepau
[vazheldakova@fedora os-intro]$ rm package.json
[vazheldakova@fedora os-intro]$ make COURSE=os-intro
[vazheldakova@fedora os-intro]$ git add .
[vazheldakova@fedora os-intro]$ git commit -am 'feat(main): make course struc
```

Рис.10.1 Настройка каталога курса

```
create mode 100644 project-personal/stage1/report/report.md
create mode 100644 project-personal/stage2/presentation/Makefile
create mode 100644 project-personal/stage2/presentation/presentation.md
create mode 100644 project-personal/stage2/report/Makefile
create mode 100644 project-personal/stage2/report/bib/cite.bib
create mode 100644 project-personal/stage2/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage2/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage2/report/report.md
create mode 100644 project-personal/stage3/presentation/Makefile
create mode 100644 project-personal/stage3/presentation/presentation.md
create mode 100644 project-personal/stage3/report/Makefile
create mode 100644 project-personal/stage3/report/bib/cite.bib
create mode 100644 project-personal/stage3/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage3/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage3/report/report.md
create mode 100644 project-personal/stage4/presentation/Makefile
create mode 100644 project-personal/stage4/presentation/presentation.md
create mode 100644 project-personal/stage4/report/Makefile
create mode 100644 project-personal/stage4/report/bib/cite.bib
create mode 100644 project-personal/stage4/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage4/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage4/report/report.md
create mode 100644 project-personal/stage5/presentation/Makefile
create mode 100644 project-personal/stage5/presentation/presentation.md
create mode 100644 project-personal/stage5/report/Makefile
create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
[vazheldakova@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.52 КиБ | 2.13 МБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно испо
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:bermuly/study_2021-2022_os-intro.git
   eee8111..1fc1802 master -> master
```

Рис.10.2 Отправка изменений в репозиторий

## Вывод:

Изучили идеологию и применение средств контроля версий. Получили практические навыки по работе с git.

## Контрольные вопросы:

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



Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.

Она предназначена для задач, когда необходимо часто вносить изменения в проект, но при этом должна сохраняться работоспособность. VCS отслеживает изменения в файлах, предоставляет возможности для создания новых и слияние существующих ветвей проекта, производит контроль доступа пользователей к проекту, позволяет откатывать исправления и определять кто, когда и какие изменения вносил в проект.

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

Основным понятием VCS является репозиторий (repository) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. В распоряжении разработчика имеется так называемая “рабочая копия” (working copy) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию (такая операция называется commit), при запуске команды `git log` можно получить список последних коммитов в обратном хронологическом порядке (историю).

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

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществляется через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. Две наиболее известные DVCS – это Git и Mercurial.

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

Разработчик работает с начальной веткой, для разных частей проекта может создать отдельные ветки. После внесения изменений разработчик

коммитит (commit) и пушит (push) их, то есть сохраняет изменения в общем хранилище.

5. Опишите порядок работы с общим хранилищем VCS.

Каждый разработчик работает над отдельной частью проекта в своей ветке или над изменениями, сделанными другими разработчиками. После внесения изменений разработчик коммитит (commit) и пушит (push) их в общее хранилище. В конце работы необходимо выполнить слияние (merge) веток, например, с начальной.

6. Каковы основные задачи, решаемые инструментальным средством git?

С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

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

`git commit`

– создание новой ветки, базирующейся на текущей:

`git checkout -b имя_ветки`

– переключение на некоторую ветку:

`git checkout имя_ветки`

– 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Первый разработчик начал верстать проект и сделал первые коммиты, в которых зафиксировал изменения в определённом файле. После того как он сделал два коммита, он захотел отправить свои изменения в удалённый репозиторий. Чтобы их передать, разработчик выполнил команду `git push`.

После чего в облаке появилось две версии проекта. То есть он отправил не только финальную версию проекта, но и все сохранённые изменения.

После пуша данные синхронизировались с удалённым репозиторием. Для того, чтобы получить эти изменения, другой разработчик выполняет команду `git pull` и получает все изменения из облака к себе на компьютер. Таким образом, состояние проекта у обоих синхронизировалось, и они могут дальше продолжить работать над ним.

#### 9. Что такое и зачем могут быть нужны ветви (branches)?

Branches (ветки) представляют собой отдельные копии проекта. Они используются для одновременной разработки отдельных частей проекта.

#### 10. Как и зачем можно игнорировать некоторые файлы при commit?

Для игнорирования файлов при commit используют файл `.gitignore`. В него записываются файлы, которые не нужны для проекта, например, временные файлы, создаваемые редакторами, системные файлы и скомпилированный код.