Отчёт по лабораторной работе №3

Markdown

Митичкина Екатерина Павловна

Оглавление

Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

Задача

Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.

Теоретическое введение:

Базовые сведения о Markdown Чтобы создать заголовок, используйте знак (#), например:

```
# This is heading 1
## This is heading 2
```

```
### This is heading 3
#### This is heading 4
```

Чтобы задать для текста полужирное начертание, заключите его в двойные звездочки: This text is `**bold**`.

Чтобы задать для текста курсивное начертание, заключите его в одинарные звездочки: This text is `*italic*`.

Чтобы задать для текста полужирное и курсивное начертание, заключите его в тройные звездочки: This is text is both `***boldand italic***`.

Блоки цитирования создаются с помощью символа >:

```
> The drought had lasted now for ten million years, and the reign of the terribl
e lizards had long since ended. Here on the Equator, in the continent which woul
d one day be known as Africa, the battle for existence had reached a new climax
of ferocity, and the victor was not yet in sight. In this barren and desiccated
land, only the small or the swift or the fierce could flourish, or even hope to
survive.
```

Неупорядоченный (маркированный) список можно отформатировать с помощью звездочек или тире:

```
- List item 1
- List item 2
- List item 3
```

Чтобы вложить один список в другой, добавьте отступ для элементов дочернего списка:

```
- List item 1
    - List item A
    - List item B
- List item 2
```

Упорядоченный список можно отформатировать с помощью соответствующих цифр:

```
1. First instruction
1. Second instruction
1. Third instruction
```

Чтобы вложить один список в другой, добавьте отступ для элементов дочернего списка:

```
1. First instruction
    1. Sub-instruction
    1. Sub-instruction
1. Second instruction
```

Синтаксис Markdown для встроенной ссылки состоит из части `[link text]`, представляющей текст гиперссылки, и части `(file-name.md)` – URL-адреса или имени файла, на который дается ссылка:

```
[link text](file-name.md)
```

Markdown поддерживает как встраивание фрагментов кода в предложение, так и их размещение между предложениями в виде отдельных огражденных блоков. Огражденные блоки кода — это простой способ выделить синтаксис для фрагментов кода. Общий формат огражденных блоков кода:

```
your code goes in here
```

Верхние и нижние индексы: H_2O записывается как `H~2~O`

2^{10} записывается как `2^10^`

Внутритекстовые формулы делаются аналогично формулам LaTeX. Например, формула $\sin^2(x) + \cos^2(x) = 1!$ запишется как `$\sin^2 (x) + \cos^2 (x) = 1$!`

Выключные формулы:

$$\sin^2(x) + \cos^2(x) = 1!$$ {#eq:eq:sin2+cos2} со ссылкой в тексте «Смотри формулу ([-@eq:eq:sin2+cos2]).» записывается как

```
$$
\sin^2 (x) + \cos^2 (x) = 1
$$ {#eq:eq:sin2+cos2}

Смотри формулу ([-@eq:eq:sin2+cos2]).
```

Обработка файлов в формате Markdown

Для обработки файлов в формате Markdown будем использовать Pandoc https://pandoc.org/. Конкретно, нам понадобится программа pandoc , pandoc-citeproc https://github.com/jgm/pandoc/releases, pandoc-crossref https://github.com/lierdakil/pandoc-crossref/releases. Преобразовать файл README.md можно следующим образом:

```
pandoc README.md -o README.pdf
```

или так

```
pandoc README.md -o README.docx
```

Можно использовать следующий Makefile

```
FILES = $(patsubst %.md, %.docx, $(wildcard *.md)) FILES += $(patsubst %.md, %.p
df, $(wildcard *.md))

LATEX_FORMAT =

FILTER = --filter pandoc-crossref

%.docx: %.md
-pandoc "$<" $(FILTER) -o "$@"

%.pdf: %.md
```

```
-pandoc "$<" $(LATEX_FORMAT) $(FILTER) -o "$@"

all: $(FILES)
@echo $(FILES)

clean:
-rm $(FILES) *~
```

Выполнение лабораторной работы

Отчет по лабораторной работе №2

Цель работы

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

Задание

– Создать базовую конфигурацию для работы с git. – Создать ключ SSH. – Создать ключ PGP. – Настроить подписи git. – Зарегистрироваться на Github. – Создать локальный каталог для выполнения заданий попредмету.

Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе несколь- ких 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 боль- шинства 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 (авто- матически 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 нескольки- ми версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные

```
cd /tmp

wget    --no-check-certificate    -q    https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-
installer.sh chmod +x gitflow-installer.sh

sudo ./gitflow-installer.sh install stable
```

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

```
sudo dnf install gh
```

классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределён- ных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

```
[epmitichkina@fedora tmp]$ git config --global user.name "Kate Mitichkina"
[epmitichkina@fedora tmp]$ git config --global user.email "mitiakata8@gmail.com"
[epmitichkina@fedora tmp]$ git config --global core.quotepath false
```

Выполнение лабораторной работы

Так как у меня уже была учетная запись на github, я пропустила пункт настройки github. Далее перешла к установке программного обеспечения.

1) Установка git-flow в Fedora Linux Для этого я запустила код

2) Установка gh в Fedora Linux

Все программное обеспечение загрузилось Следящий пункт базовая настройка git

Для начала я задала имя и email репозиторию:

Настроила utf-8 в выводе сообщений git, задала имя начальной ветки, параметр autocrlf и safecrlf

```
[epmitichkina@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/epmitichkina/.ssh/id_ed25519): /home/epmitichkina/.ssh/id_ed
19
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/epmitichkina/.ssh/id_ed25519
Your public key has been saved in /home/epmitichkina/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:xMZJAe/d9TDCdJUQCQPbLsvZqJEWpPRku2mbjE8zAOw epmitichkina@fedora
The key's randomart image is:
+--[ED25519 256]--+
|      ..oo.ooo=.o|
|  .    = .oo.o . |
|   o . +B. .o +  |
|  . o *+.... o + |
|   E o +S....   .|
|      . * *      |
|       @ = .     |
|      * B        |
|     ..*         |
+----[SHA256]-----+
```

После этого создала ключи ssh

–по алгоритму rsa с ключём размером 4096 бит:

–по алгоритму ed25519:

```
[epmitichkina@fedora tmp]$ git config --global core.quotepath false
[epmitichkina@fedora tmp]$ git config --global init.defaultBranch master
[epmitichkina@fedora tmp]$ git config --global core.autocrlf input
[epmitichkina@fedora tmp]$ git config --global core.safecrlf warn
```

```
[epmitichkina@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/epmitichkina/.ssh/id_rsa): /home/epmitichkina/.ssh/id_rsa
Created directory '/home/epmitichkina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/epmitichkina/.ssh/id_rsa
Your public key has been saved in /home/epmitichkina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:+YpZEwmuyyi9F+t94tc/NWDOy0VX6mzgOHbpN60dhtw epmitichkina@fedora
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                .|
|      .        ..|
|     . . o  + o .|
|      . S  * B . |
|    ..   o+ B X  |
| .  .o  oo.= B E |
|. .ooo.+oo. = +.o|
| .o++.=+.  ..o.o.|
+----[SHA256]-----+
```

Следующий пункт был создать ключи pgp

```
[epmitichkina@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/epmitichkina/.gnupg'
gpg: создан щит с ключами '/home/epmitichkina/.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: Kate47
Адрес электронной почты: mitiakata8@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Kate47 <mitiakata8@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/epmitichkina/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ C699DE047EDF4CDA помечен как абсолютно доверенный
gpg: создан каталог '/home/epmitichkina/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/epmitichkina/.gnupg/openpgp-revocs.d/D0E416EA2279C17E9B46801C
E047EDF4CDA.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-20 [SC]
      D0E416EA2279C17E9B46801CC699DE047EDF4CDA
uid                      Kate47 <mitiakata8@gmail.com>
sub   rsa4096 2022-04-20 [E]
```

Дальше нужно было добавить PGP ключа в GitHub
Вывила список ключей и скопировала отпечаток приватного ключа

```
[epmitichkina@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/epmitichkina/.gnupg/pubring.kbx
------------------------------------
sec   rsa4096/C699DE047EDF4CDA 2022-04-20 [SC]
      D0E416EA2279C17E9B46801CC699DE047EDF4CDA
uid                 [  абсолютно ] Kate47 <mitiakata8@gmail.com>
ssb   rsa4096/79C81C5CA36F0FB6 2022-04-20 [E]
```

Скопировала сгенерированный PGP ключ в буфер обмена:

```
[epmitichkina@fedora tmp]$ gpg --armor --export C699DE047EDF4CDA | xclip -sel cli
```

И вставила полученный код на github Дальше настроила автоматические подписи коммитов git

```
[epmitichkina@fedora tmp]$ git config --global user.signingkey C699DE047EDF4CDA
[epmitichkina@fedora tmp]$ git config --global commit.gpgsign true
[epmitichkina@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

После нужно было авторизироваться

```
[epmitichkina@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: 0B84-F621
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 ate4
```

Следующие создание репозитория курса на основе шаблона

```
[epmitichkina@fedora tmp]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[epmitichkina@fedora tmp]$ cd  ~/work/study/2022-2023/"Операционные системы"
```

```
[epmitichkina@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository ate4/study_2022-2023_os-intro on GitHub
```

И последнее это настройка каталога курса Перехожу в нужный каталог и удаляю не нужный файл, далее создаю нужный каталог и отправляю нужный файл на сервер

```
[epmitichkina@fedora Операционные системы]$ cd study_2022-2023_os-intro/
[epmitichkina@fedora study_2022-2023_os-intro]$ rm package.json
[epmitichkina@fedora study_2022-2023_os-intro]$ make COURSE=os-intro
[epmitichkina@fedora study_2022-2023_os-intro]$ git add
Ничего не указано, ничего не добавлено.
подсказка: Maybe you wanted to say 'git add .'?
подсказка: Turn this message off by running
подсказка: "git config advice.addEmptyPathspec false"
[epmitichkina@fedora study_2022-2023_os-intro]$ git add COURSE=os-intro
fatal: спецификатор пути «COURSE=os-intro» не соответствует ни одному файлу
[epmitichkina@fedora study_2022-2023_os-intro]$ make COURSE=os-intro
make: Цель «all» не требует выполнения команд.
[epmitichkina@fedora study_2022-2023_os-intro]$ git add .
[epmitichkina@fedora study_2022-2023_os-intro]$ git commit -am 'feat(main): make course structure'
[master 27c51dc] feat(main): make course structure
 149 files changed, 16590 insertions(+), 14 deletions(-)
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/presentation.md
 create mode 100644 labs/lab01/report/Makefile
 create mode 100644 labs/lab01/report/bib/cite.bib
 create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab01/report/report.md
 create mode 100644 labs/lab02/presentation/Makefile
 create mode 100644 labs/lab02/presentation/presentation.md
 create mode 100644 labs/lab02/report/Makefile
 create mode 100644 labs/lab02/report/bib/cite.bib
 create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab02/report/report.md
 create mode 100644 labs/lab03/presentation/Makefile
 create mode 100644 labs/lab03/presentation/presentation.md
 create mode 100644 labs/lab03/report/Makefile
 create mode 100644 labs/lab03/report/bib/cite.bib
 create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab03/report/report.md
 create mode 100644 labs/lab04/presentation/Makefile
 create mode 100644 labs/lab04/presentation/presentation.md
 create mode 100644 labs/lab04/report/Makefile
 create mode 100644 labs/lab04/report/bib/cite.bib
 create mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab04/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab04/report/report.md
 create mode 100644 labs/lab05/presentation/Makefile
 create mode 100644 labs/lab05/presentation/presentation.md
 create mode 100644 labs/lab05/report/Makefile
 create mode 100644 labs/lab05/report/bib/cite.bib
 create mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab05/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab05/report/report.md
 create mode 100644 labs/lab06/presentation/Makefile
 create mode 100644 labs/lab06/presentation/presentation.md
```

```
[epmitichkina@fedora Операционные системы]$ git clone --recursive https://github.com/ate4/study_2022-2023_os-intro
Клонирование в «study_2022-2023_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), 12.48 КиБ | 277.00 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/presenta
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/epmitichkina/work/study/2022-2023/Операционные системы/study_2022-2023_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 | 1.15 МиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/home/epmitichkina/work/study/2022-2023/Операционные системы/study_2022-2023_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 | 1.01 МиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eaebb7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
```

```
 create mode 100644 structure
[epmitichkina@fedora study_2022-2023_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), 266.53 КиБ | 4.76 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/ate4/study_2022-2023_os-intro
   9094309..27c51dc  master -> master
[epmitichkina@fedora study_2022-2023_os-intro]$
```

Выводы

В результате работы изучила идеологию и применение средств контроля версий. Освоила умения по работе с git.

Ответы на контрольные вопросы

1) Version Control System — программное обеспечение для облегчения работы с изменяющейся информацией. 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

2) 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3) Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4) 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" и настроив utf-8 в выводе сообщенийgit: git config —global quotepath false Для инициализации

локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке: cd mkdir tutorial cd tutorial git init

5) Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C"Имя Фамилия work@mail" Ключи сохраняться в каталоге~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id_rsa.pub | xclip -sel clip вставляем ключ в появившееся на сайте поле.

6) У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) Основные команды git: Наиболее часто используемые команды 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–просмотртекущих изменения: 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 натекущей: 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 стекущим 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) Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): git add hello.txt git commit -am' Новый файл

9) Проблемы, которые решают ветки git: • нужно постоянно создавать архивы с рабочим кодом • сложно "переключаться" между архивами • сложно перетаскивать изменения между архивами • легко что-то напутать или потерять

10) Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл gitignore с помощью сервисов. Для этого сначала нужно получить

списки меняющихся 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

Вывод

В результате работы научилась оформлять отчёты с помощью легковесного языка разметки Markdown.