**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Операционные системы*

Студент:

*Гайсина Алина Ринатовна*

Группа:

НКНбд-02-21

**МОСКВА**

2022 г.

**Цель работы:**

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

**Ход работы:**

1. Создала учетную запись на GitHub.



Рисунок 1. Учетная запись на GitHub.

1. Вручную установила программное обеспечение git-flow.

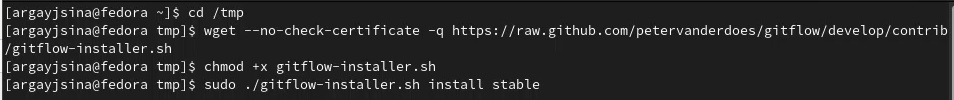


Рисунок 2. Установка git-flow.

1. Установила gh.



Рисунок 3. Установка gh.

1. Задала имя и email владельца репозитория.

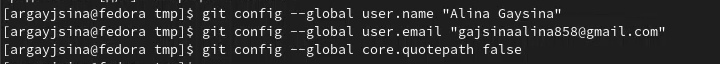


Рисунок 4. Имя и email.

1. Настроила utf-8 в выводе сообщений git.

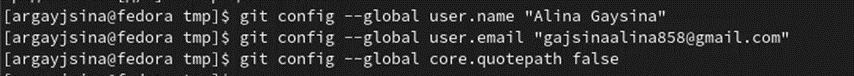


Рисунок 5. Настройка utf-8.

1. Настроила верификацию и подписание коммитов git и задала имя начальной ветки (master).



Рисунок 6. Коммиты и начальная ветка.

1. Создала ключ ssh.

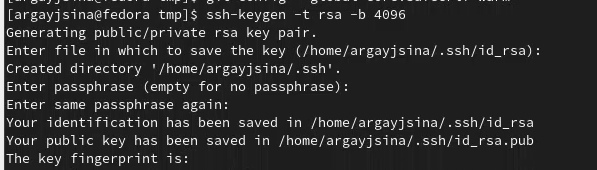


Рисунок 7 1

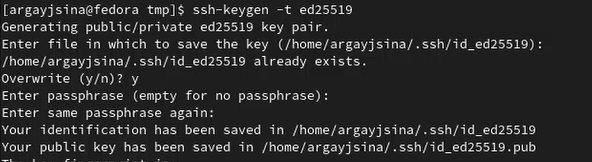


Рисунок 7 2

1. Создала ключ pgp и добавила его в GitHub.

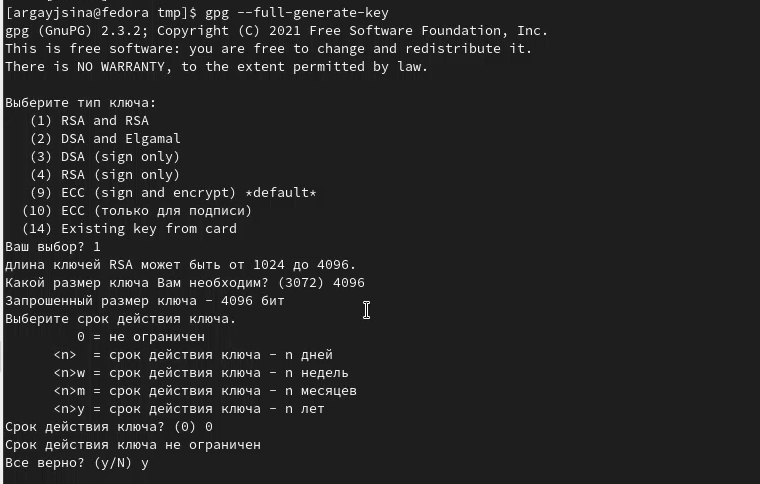


Рисунок 8 1

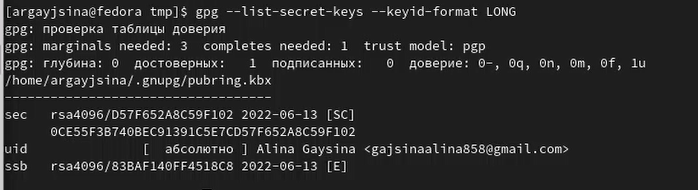


Рисунок 8 2



Рисунок 8 3



Рисунок 8 4

1. Настроила автоматические подписи коммитов git.



Рисунок 9

1. Настроила gh.

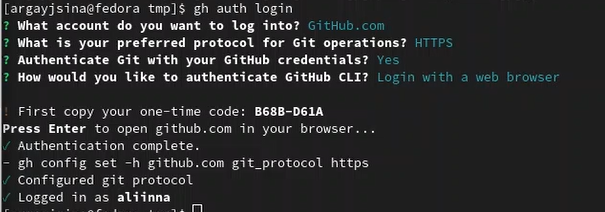


Рисунок 10

1. Создала репозиторий курса на основе шаблона.



Рисунок 11 1



Рисунок 11 2



Рисунок 11 3

1. Настроила каталоги курса.



Рисунок 12 1



Рисунок 12 2

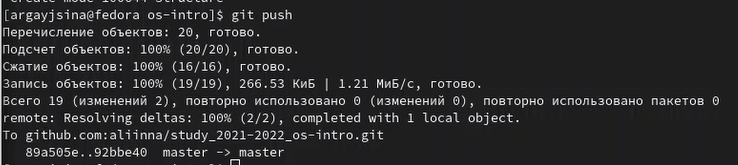


Рисунок 12 3

**Выводы:**

Изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

**Контрольные вопросы:**

1. 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. Понятия:
   * Хранилище – репозиторий - место хранения всех версий и служебной информации.
   * Commit — это команда для записи индексированных изменений в репозиторий.
   * История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.
   * Рабочая копия – текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Отличия централизованных и децентрализованных VCS:
   * Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример – Subversion.
   * Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев. Пример – Git.
4. Правки в рабочей области появляются только если сам пользователь отправляет их на сервер.
5. Если хранилище общее, то в рабочую копию каждого, кто работает над проектом, приходят изменения, отправленные на сервер одним из команды. Рабочая правка каждого может изменяться вне зависимости от того, делает ли конкретный человек правки или нет.
6. Хранить информацию обо всех изменениях в вашем коде и обеспечивать удобство командной работы над кодом.
7. 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
   * просмотр текущих 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
   * удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов
   * 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 на текущей: 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: 1 git push origin имя\_ветки
   * слияние ветки с текущим деревом: 1 git merge —no-ff имя\_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. Удаленные репозитории: git remote, локальные – git status.
9. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл . Временно игнорировать изменения в файле можно командой git update-index-assumeunchanged.