Российский университет дружбы народов  
Факультет Физико-математический и естественных наук

Операционные системы

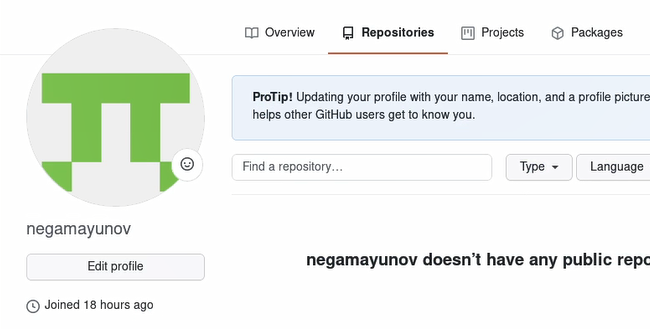
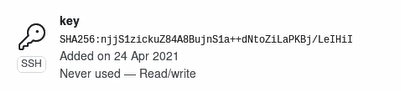
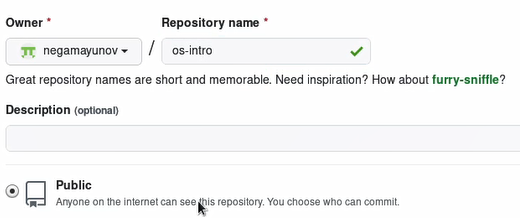
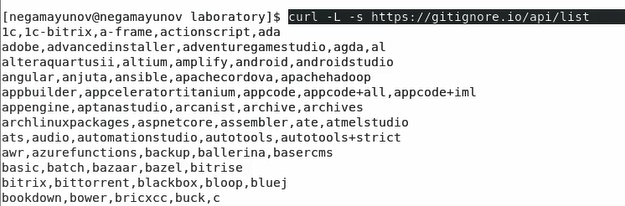
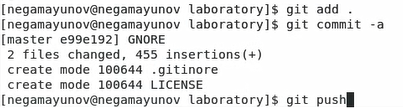
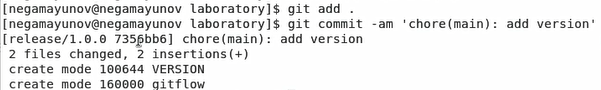
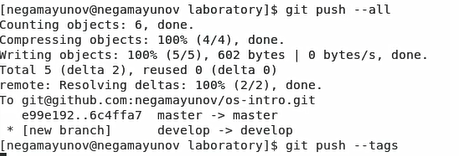
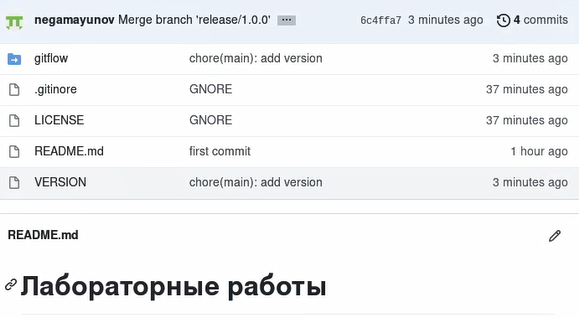
**Лабораторная работа №2**

Выполнил Гамаюнов Никита Ефимович,  
1032201719, НПМбд-01-20

*Москва, 2021 год*

**Цель работы:** изучить идеологию и применение средств контроля версий

**Результаты выполнения задания (последовательность выполнения):**

1. Создал учетную запись на GitHub:
2. Настроил систему контроля версий git:
3. Сгенерировал ssh-ключ:
4. Добавил его в настройках учетной записи на GitHub:
5. Создал структуру каталога лабораторных работ с помощью mkdir:
6. Создал репозиторий на GitHub. Назвал его os-intro: 
7. Перешёл в каталог laboratory:
8. Инициализировал системы git:
9. Создал заготовку для файла README.md:
10.  Создал первый коммит и отправил его на GitHub:
11. Добавил файл лицензии:
12.  Просмотрел список имеющихся шаблонов игнорируемых файлов:
13.  Загрузил шаблон для С:
14.  Добавил новые файлы, отправил на GutHub:
15.  Инициализировал git-flow:
16.  Префикс для ярлыков установил в v:
17.  Находясь на ветке develop, создал релиз с версией 1.0.0:
18.  Записал версию:
19.  Добавил в индекс:
20.  Залил релизную ветку в основную ветку:
21. Отправил данные на GitHub:
22.  Проверил созданный релиз на GitHub:

**Вывод:** я изучил идеологию и применение средств контроля версий.

**Ответы на контрольные вопросы:**

1. Система контроля версий (VCS) – ПО, облегчающее реализацию удалённой работы нескольких пользователей над одним проектом.
2. Хранилище (репозиторий) – это буквально хранилище для всей информации, которая используется в проекте (код, файлы, изменения в версиях и т.д.)

Commit – команда, позволяющая сохранить все добавленные изменения.

История – сортированный по времени список всех когда-либо внесённых изменений.

Рабочая копия – копия удалённого репозитория, расположенная на устройстве пользователя. Именно её можно изменять и синхронизировать с репозиторием через commit.

1. Централизованная (классическая) VCS представляет собой один репозиторий для хранения файлов. Самые известные примеры: CVS, Subversion.

Децентрализованная (распределённая) VCS не предполагает обязательного наличия центрального репозитория, её суть в том, что файлы хранятся не на одном сервере, а на устройствах всех участников проекта. Примеры: Git, Bazaar, Mercurial.

1. При единоличной работе с хранилищем на устройстве создаётся локальный репозиторий, в файлы вносятся необходимые изменения, которые через commit отправляются уже в удалённый репозиторий.
2. При работе с общим хранилищем обычно файлы скачиваются из удалённого репозитория, создаётся новая ветка, на которой в файлы так же вносятся и сохраняются с помощью commit нужные изменения. После этого ветка отправляется в удалённый репозиторий, где сливается с основной веткой.
3. Основные задачи git – обеспечение удобства работы с версиями и хранения информации (кода, файлов, etc.)
4. – 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 –   
   – 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   
   – 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: git push origin имя\_ветки   
   – слияние ветки с текущим деревом: 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
5. С локальным репозиторием git можно использовать для работы с личными проектами, - система контроля версий позволяет удобно систематизировать файлы и экспериментировать, ведь в случае неудачи можно откатиться на более раннюю версию.

С удалённым репозиторием удобно осуществлять групповые работы, - помимо возможного отката на ранние версии здесь можно следить за работой каждого из участников, а значит, эффективнее распределять задачи и корректировать проект.

1. Ветвление – это способ хранения проекта. В удалённом репозитории имеется главная ветка – master branch, и каждые его участник, начиная работу, создаёт на ней свою, отдельную ветку. После окончания работы ветку можно слить с главной, перенеся все изменения на неё, или оставить код на проверку другими участниками или администраторами проекта.  
   Ветвление удобно, потому что каждый участник проекта может работать обособленно и при этом, если что, сразу загружать себе данные от других пользователей удалённого репозитория.
2. Некоторые файлы не нужно добавлять в проект, - например, файлы редакторов, остаточные файлы и прочий мусор. С помощью gitignore можно удобно выбрать шаблон игнорирования и автоматически избавить себя от необходимости «чистить» репозиторий от ненужный файлов. Список шаблонов можно просмотреть по команде curl -L -s https://www.gitignore.io/api/list.  
   После этого можно загрузить нужный шаблон, например, для C: curl -L -s https://www.gitignore.io/api/c >> .gitignore. После этого ненужные файлы, оставляемые редакторами языка C, объектные файлы и т.д. не будут попадать в конечную версию проекта.