# Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

**Лабораторная работа 2**

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032193844

Саидов Ахият Магомадович

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

# Цель работы

Мы изучим идеологию и применение средств контроля версий, также мы освоим умения по работе с git.

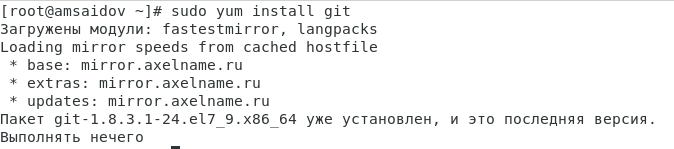
# Начало работы

## Настроим github.

Для этого предварительно создадим учетную страницу на сайте github.com.

Мы создали УС: <https://github.com/Akhiyat>; и заполнили необходимые данные.  
Установим программное обеспечение.

Чтобы установить git из стандартного репозитория CentOS, используем менеджер пакетов yum.



*Рисунок 1*

Примечание: ранее уже был установлен git, исходя из этого, нам указывается, что устанавливать нечего.

## Базовые настройки git.

Зададим имя и email владельца репозитория

fig:

*Рисунок 2*

Настроим utf-8 в выводе сообщения git

fig:

*Рисунок 3*

## Настроим верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть ее master).

Параметр autocrlf.

Параметр safecrlf.

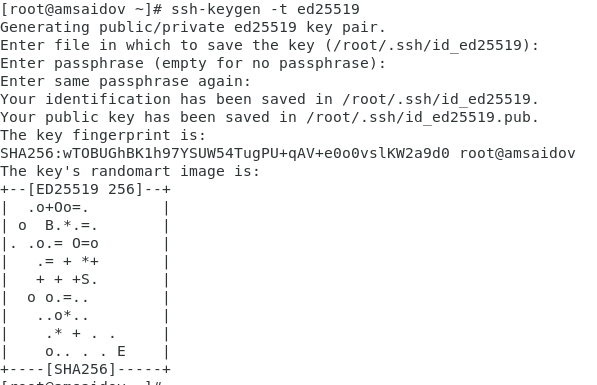
fig:

Создадим ключи SSH.

По алгоритму rsa с ключем размеров 4096 бит.

По алгоритму ed25519.

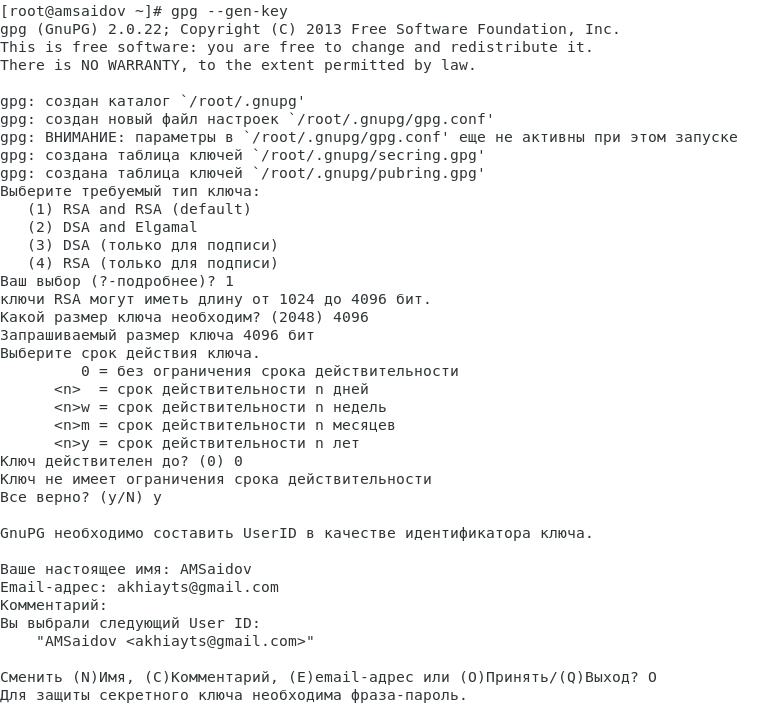




*Рисунок 5*

## Создадим ключи GPG.

Генерируем ключ.



– Из предложенных опций выбираем: – тип RSA and RSA; – размер 4096; – выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда). – GPG запросит личную информацию, которая сохранится в ключе: – Имя (не менее 5 символов). – Адрес электронной почты. – Комментарий.

*Рисунок 7*

## Добавление GPG ключа в GitHub.

Скопируем наш сгенерированный PGP ключ в буфер обмена.



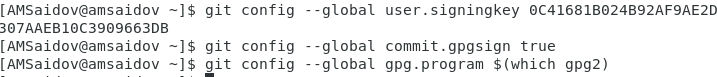
*Рисунок 8*

Вставим ключ в GitHub.



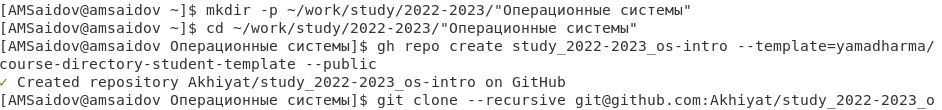
*Рисунок 9*

Используем введенный email, укажем Git применять его при подписи коммитов.



*Рисунок 10*

Создадим репозиторий курса.



*Рисунок 11*

Перейдем в каталог курса.

fig:

*Рисунок 12*

Удалим лишние файлы.

fig:

*Рисунок 13*

Создадим необходимые каталоги.

fig:

*Рисунок 14*

Отправим файлы на сервер.

fig:

fig:

fig:

*Рисунок 15*

# Вывод

Мы изучили идеологию и применение средств контроля версий. Мы освоили умения по работе с git.

# Контрольные вопросы

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
   * Хранилище (репозиторий) – это система, которая обеспечивает хранение всех существовавших версий файлов.
   * Commit - запись изменений.
   * История - список предыдущих изменений.
   * Рабочая копия – копия файла, с которой непосредственно ведётся работа (находится вне репозитория) С помощью коммитов изменения, внесённые в рабочую копию, заносятся в хранилище. Благодаря истории можно отследить изменения, вносимые в репозиторий. Перед началом работы рабочую копию можно получить из одной из версий, хранящихся в репозитории.
2. В централизованных СКВ все файлы хранятся в одном репозитории, и каждый пользователь может вносить изменения. В децентрализованных их несколько, и они могут обмениваться изменениями между собой, а центрального репозитория может не существовать вообще. Среди классических (т.е. централизованных) VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.
3. Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).
4. Аналогично единоличной работе, но также можно объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику.
5. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Git позволяет создавать локальные репозитории и вносить в них изменения, а также работать с удалёнными репозиториями.
   * 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 5)просмотр текущих изменения: git diff
   * сохранение текущих изменений: а)добавить все изменённые и/или созданные файлы и/или каталоги: git add . б)добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена*файлов в)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: а)сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита' б)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 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
6. Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am “file.txt was added” Отправим изменения в удалённый репозиторий: git push
7. СКВ могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Это удобно при работе над одним проектом нескольких человек, или если вносимые на каждой из ветвей изменения будут разительно отличаться (например, создание программ с разным функционалом на базе одного интерфейса).
8. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять впоследствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы