

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
ИМЕНИ ПАТРИСА ЛУМУМБЫ

Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Операционные системы

Студент: Благова Полина

Группа: НПМбв-19

МОСКВА

2023 г.

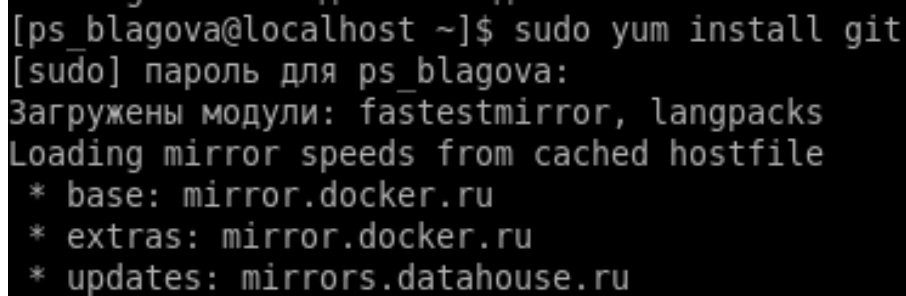
Цель работы:

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.
- Создать ключи SSH и PGP.

Описание результатов выполнения задания:

- Создать базовую конфигурацию для работы с git.

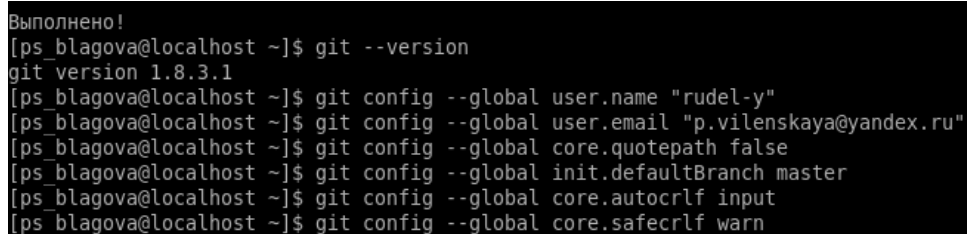
Устанавливаю программный пакет для работы с git



```
[ps_blagova@localhost ~]$ sudo yum install git
[sudo] пароль для ps_blagova:
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.docker.ru
 * extras: mirror.docker.ru
 * updates: mirrors.datahouse.ru
```

Рис. 1. Загрузка пакета git

Провожу базовую настройку: имя владельца репозитория, почта владельца, настройка utf-8 в выводе сообщений git, создание master ветки, верификация и подпись коммитов, параметры autocrlf и safecrlf.



```
Выполнено!
[ps_blagova@localhost ~]$ git --version
git version 1.8.3.1
[ps_blagova@localhost ~]$ git config --global user.name "rudel-y"
[ps_blagova@localhost ~]$ git config --global user.email "p.vilenskaya@yandex.ru"
[ps_blagova@localhost ~]$ git config --global core.quotepath false
[ps_blagova@localhost ~]$ git config --global init.defaultBranch master
[ps_blagova@localhost ~]$ git config --global core.autocrlf input
[ps_blagova@localhost ~]$ git config --global core.safecrlf warn
```

Рис. 2. Базовая настройка git

– Создать ключ SSH.

По алгоритму rsa с ключом размером 4096 бит и по алгоритму ed25519

```
[ps_blagova@localhost ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ps_blagova/.ssh/id_rsa): ssh-keygen -t ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh-keygen -t ed25519.
Your public key has been saved in ssh-keygen -t ed25519.pub.
The key fingerprint is:
SHA256:450JL6c19jMd6Gd/Wat7d2YcsSov+hje/0yoQWJR8x8 ps_blagova@localhost.localdomain
The key's randomart image is:
+----[RSA 4096]-----+
|
|   o
|  . o
|   . E
|   . ...
|  So o .o|
|  o.++ . .o.|
|  . 0o o o.o=|
|  .+o+Bo*.o+B|
|  .+.++0=***o|
+-----[SHA256]-----+
```

Рис. 3. Создание ключа ssh

– Создать ключ PGP.

Генерирую ключ. Из предложенных опций выбираю:

- 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.
- При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
- Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
[ps_blagova@localhost ~]$ gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 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/ps_blagova/.gnupg'
gpg: создан новый файл настроек '/home/ps_blagova/.gnupg/gpg.conf'
gpg: ВНИМАНИЕ: параметры в '/home/ps_blagova/.gnupg/gpg.conf' еще не активны при этом запуске
gpg: создана таблица ключей '/home/ps_blagova/.gnupg/secring.gpg'
gpg: создана таблица ключей '/home/ps_blagova/.gnupg/pubring.gpg'
Выберите требуемый тип ключа:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
Ваш выбор (?-подробнее)? 1
```

Рис. 4. Создание ключа gpg

```
ключи RSA могут иметь длину от 1024 до 4096 бит.
Какой размер ключа необходим? (2048) 4096
Запрашиваемый размер ключа 4096 бит
Выберите срок действия ключа.
  0 = без ограничения срока действительности
  <n> = срок действительности n дней
  <n>w = срок действительности n недель
  <n>m = срок действительности n месяцев
  <n>y = срок действительности n лет
Ключ действителен до? (0) 0
Ключ не имеет ограничения срока действительности
Все верно? (y/N) y

GnuPG необходимо составить UserID в качестве идентификатора ключа.

Ваше настоящее имя: rudel-y
Email-адрес: p.vilenskaya@yandex.ru
Комментарий: hello_world
Вы выбрали следующий User ID:
  "rudel-y (hello_world) <p.vilenskaya@yandex.ru>"
```

Рис. 5. Генерация ключа gpg

Вывожу список ключей и копирую отпечаток приватного ключа:

```
[ps_blagova@localhost ~]$ gpg --list-secret-keys --keyid-format LONG
/home/ps_blagova/.gnupg/secring.gpg
-----
sec 4096R/A9E44F01565E8236 2023-06-11
uid                                rudel-y (hello_world) <p.vilenskaya@yandex.ru>
ssb 4096R/33486ADBEE5C3719 2023-06-11
```

Рис. 6. Отпечаток приватного ключа.

Вывожу публичный ключ и копирую его.

```
[ps_blagova@localhost ~]$ gpg --armor --export 33486ADBEE5C3719
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGsFyFkBEADhEgKSqS1HYMJU8014J1f311LrQ2CIrjuDr8HX987L46LpfnWq
5+td3hVFXGKv3R8KCv1z00XdPfi8+9Eahh+qwnq5yW05HNPY6bfBJCdaCqNxXHjc
zLEl1+RUlaJtuXmTQmGrmZjuorlikPjsyeZ9cgtMwtX1Wj4fDD/2gXFRIRXl1Iq
M6Zm3IM+hUWlftGc8nNnMWG4K5i26BLHTijy0CHBlomD20QWbpef9Pa+mKxflcb1
28XGpt0R4xxpVEqLXzmk/+7QwYovrhBnlzSFQfDKPbGNYhs9miVnwIlUGU3aLLPU
2PrKAFdN11Swwk71SaC9qbE76MBSsHmxNpqJf4ELK4fLJ01y/oUZDvKm0XyLG91z
NNNKxaEFURZtx4GMc5rwy011+DHRHuhFSI3tHPKGi3XeMbMp3PTjel8XuVwjMgwk
lCNC5avNc3hyfr0MqpxMNnc6b+ipT2LR0vNBmcBWIchW2lyMmdpqs9MnjZFFV6w
```

Рис. 7. Публичный ключ gpg

Ввожу публичный ключ в github.

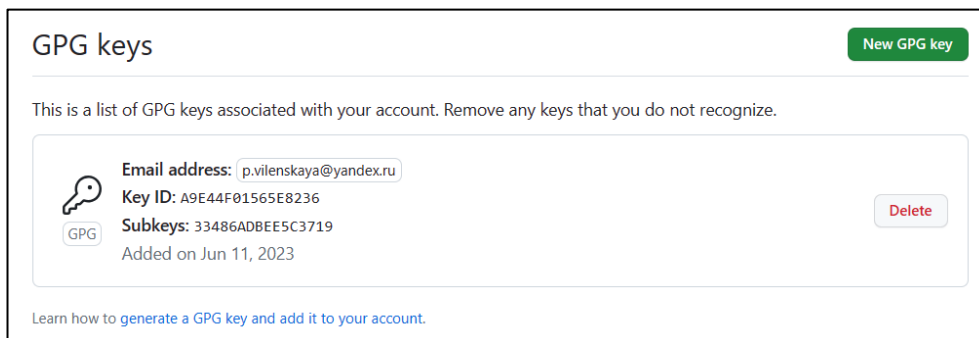


Рис. 8. Добавление публичного ключа gpg в github

– Настроить подписи git.

Используя введённый email, указываю Git, чтобы применять его при подписи КОММИТОВ

```
[ps_blagova@localhost ~]$ git config --global user.signingkey 33486ADBEE5C3719
[ps_blagova@localhost ~]$ git config --global commit.gpgsign true
[ps_blagova@localhost ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 9. Настройка автоматических подписей коммитов git

- Создать локальный каталог для выполнения заданий по предмету
- Создаю репозиторий на своем аккаунте в github для заданий по предмету.

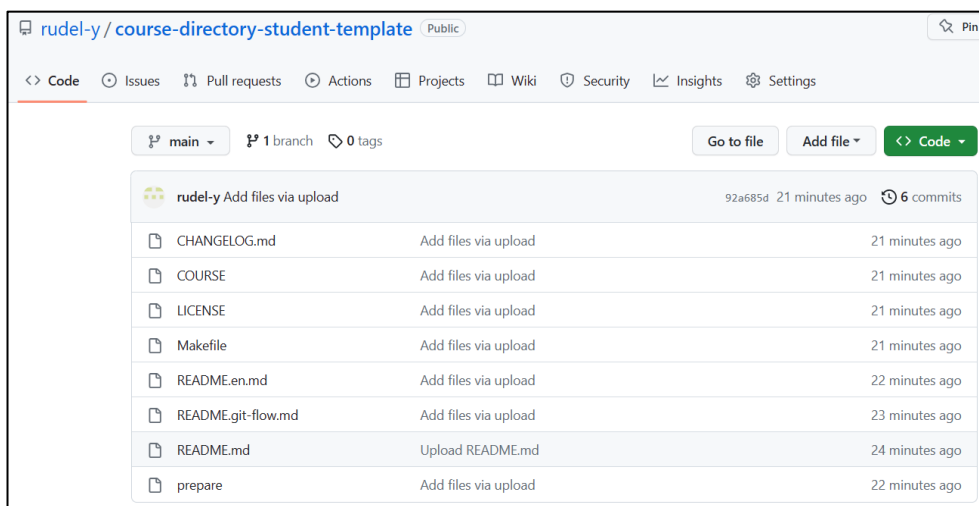


Рис. 10. Репозиторий для выполнения заданий.

Выводы:

- Были изучены идеология и применение средств контроля версий.
- Были освоены умения по работе с git.
- Были созданы ключи SSH и PGP.

Контрольные вопросы:

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

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

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.

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

В случае с централизованной VCS репозиторий хранится на одном сервере, и все разработчики работают с ним. 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.

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

Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).

5. Опишите порядок работы с общим хранилищем VCS.

Аналогично единоличной работе, но также можно объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику.

6. Каковы основные задачи, решаемые инструментальным средством git?

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.

7. Назовите и дайте краткую характеристику командам git.

- 1) Создание основного дерева репозитория: git init
- 2) Получение обновлений (изменений) текущего дерева из центрального репозитория: git pull
- 3) Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push
- 4) Просмотр списка изменённых файлов в текущей директории: git status
- 5) Просмотр текущих изменений: git diff
- 6) 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`

7) 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`

8) Создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки`

9) 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)

10) Отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`

11) Слияние ветки с текущим деревом: `git merge --no-ff имя_ветки`

12) Удаление ветки:

а) Удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`

б) Принудительное удаление локальной ветки: `git branch -D имя_ветки`

в) Удаление ветки с центрального репозитория: `git push origin :имя_ветки`

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

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`

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

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).

10. Как и зачем можно игнорировать некоторые файлы при commit

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