

Отчёт

Лабораторная работа № 2

Приходько Иван Иванович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	15
5	Ответы на контрольные вопросы	16

Список иллюстраций

3.1	Установка git	7
3.2	Установка gh	7
3.3	Указание имени и почты	7
3.4	Настройки git	8
3.5	Создание RSA ключа	8
3.6	Создание ключа по алгоритму ed25519	9
3.7	Генерация PGP ключа	10
3.8	Генерация PGP ключа	11
3.9	Список pgr ключей	11
3.10	Копирование PGP ключа	11
3.11	Вставка ключа на Github	12
3.12	Настройка автоматических подписей	12
3.13	Авторизация в gh	13
3.14	Создание рабочего пространства	13
3.15	Создание репозитория	13
3.16	Удаление ненужных файлов и работа с рабочим пространством . .	14
3.17	Добавление папки для отправки	14
3.18	Добавление коммита	14
3.19	Пуш	14

Список таблиц

1 Цель работы

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

2 Задание

Создать базовую конфигурацию для git. Создать ключ SSH и PGP. Зарегистрироваться на Github. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

Для начала установим git (рис. 3.1).

```
[prihodko_ivan_228@prihodkoIvan ~]$ sudo dnf install git
[sudo] пароль для prihodko_ivan_228:
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates          100% | 118.2 KiB/s | 25.0 KiB | 00m00s
Fedora 41 - x86_64 - Updates          100% | 530.7 KiB/s | 2.5 MiB | 00m05s
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.
```

Рис. 3.1: Установка git

Далее установим gh (рис. 3.2).

```
[prihodko_ivan_228@prihodkoIvan ~]$ sudo dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет Арх. Версия Репозиторий Размер
Установка:
gh x86_64 2.65.0-1.fc41 updates 42.6 MiB

Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] gh-0:2.65.0-1.fc41.x86_64 100% | 583.9 KiB/s | 10.3 MiB | 00m18s
```

Рис. 3.2: Установка gh

Далее зададим имя владельца репозитория и его почту (рис. 3.3).

```
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global user.name "SunHermit67"
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global user.email "33133502saq@gmail.com"
[prihodko_ivan_228@prihodkoIvan ~]$
```

Рис. 3.3: Указание имени и почты

Установим имя начальной ветки, параметр autocrlf и safecrlf (рис. 3.4).

```
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global core.quotepath false
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global init.defaultBranch master
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global core.autocrlf input
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global core.safecrlf warn
[prihodko_ivan_228@prihodkoIvan ~]$
```

Рис. 3.4: Настройки git

Создадим RSA ключ размером 4096 (рис. 3.5).

```
[prihodko_ivan_228@prihodkoIvan ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/prihodko_ivan_228/.ssh/id_rsa):
Created directory '/home/prihodko_ivan_228/.ssh'.
Enter passphrase for "/home/prihodko_ivan_228/.ssh/id_rsa" (empty for no passphrase)
:
Enter same passphrase again:
Your identification has been saved in /home/prihodko_ivan_228/.ssh/id_rsa
Your public key has been saved in /home/prihodko_ivan_228/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ycULA0AhHwYpG88LRyCrE6NbCxoYoiglgR/jfSWtv2Q prihodko_ivan_228@prihodkoIvan
The key's randomart image is:
+---[RSA 4096]-----+
|oo+*o..|
|=,==...o.|
|*Xo+. +o o|
|X=*.. o. =.|
|@o.. .S.|
|o*.. E|
|o. o.|
|.|
|.|
+-----[SHA256]-----+
```

Рис. 3.5: Создание RSA ключа

Далее создадим ключ по алгоритму ed25519 (рис. 3.6).


```

[prihodko_ivan_228@prihodkoIvan ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/prihodko_ivan_228/.ssh/id_ed25519):
Enter passphrase for "/home/prihodko_ivan_228/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/prihodko_ivan_228/.ssh/id_ed25519
Your public key has been saved in /home/prihodko_ivan_228/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:9Bb4yxecS/ZhXjOxrRRKuMc0UZidESmv17fdLCPGV84 prihodko_ivan_228@prihodkoIvan
The key's randomart image is:
+--[ED25519 256]--+
|          ++=    |
|      .  +.+.    |
|     o  .  o    |
|    . o + o.    |
|   S = X. . o   |
|    o O.*o.*    |
|   +.*.oXO|    |
|    o+.+*E|    |
|   . o.o. |    |
+-----[SHA256]-----+

```

Рис. 3.6: Создание ключа по алгоритму ed25519

Создадим PGP ключ. Выбираем тип «RSA and RSA», на 4096 бит и срок неограничен. Далее вводим свои данные и генерируем ключ (рис. 3.7, 3.8).

```
[prihodko_ivan_228@prihodkoIvan ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; Copyright (C) 2024 g10 Code GmbH
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/prihodko_ivan_228/.gnupg'
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: Ivan Prihodko
Адрес электронной почты: 33133502saq@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Ivan Prihodko <33133502saq@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.
```

Рис. 3.7: Генерация PGP ключа

```

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/prihodko_ivan_228/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/prihodko_ivan_228/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/prihodko_ivan_228/.gnupg/openpgp-revocs.d/DC
96B8229A9751DE67F01EB577C0525E7AE97F22.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2025-03-02 [SC]
       DC96B8229A9751DE67F01EB577C0525E7AE97F22
uid           Ivan Prihodko <33133502saq@gmail.com>
sub   rsa4096 2025-03-02 [E]

[prihodko_ivan_228@prihodkoIvan ~]$ █

```

Рис. 3.8: Генерация PGP ключа

Выводим список pgr ключей (рис. 3.9).

```

[prihodko_ivan_228@prihodkoIvan ~]$ 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
[keyboard]
-----
sec   rsa4096/77C0525E7AE97F22 2025-03-02 [SC]
       DC96B8229A9751DE67F01EB577C0525E7AE97F22
uid           [ абсолютно ] Ivan Prihodko <33133502saq@gmail.com>
ssb   rsa4096/5D606D9F1BBC711C 2025-03-02 [E]

```

Рис. 3.9: Список pgr ключей

Копируем наш ключ в буфер обмена (рис. 3.10).

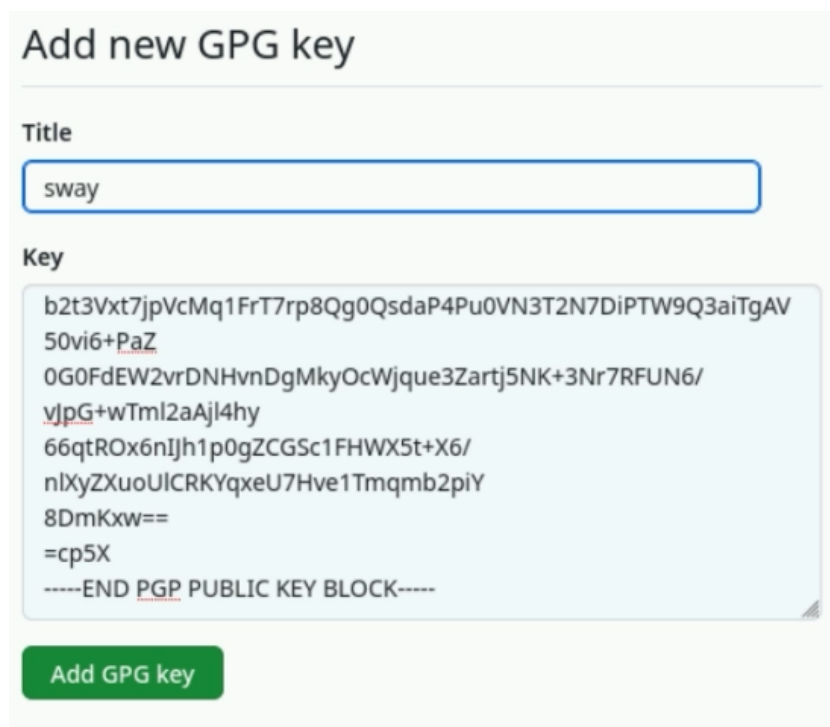
```

[prihodko_ivan_228@prihodkoIvan ~]$ gpg --armor --export 33133502saq@gmail.com | xcl
ip -sel clip

```

Рис. 3.10: Копирование PGP ключа

Добавляем наш ключ на Github (рис. 3.11).



Add new GPG key

Title

sway

Key

```
b2t3Vxt7jpVcMq1FrT7rp8Qg0QsdaP4Pu0VN3T2N7DiPTW9Q3aiTgAV
50vi6+PaZ
0G0FdEW2vrDNHvnDgMkyOcWjque3Zartj5NK+3Nr7RFUN6/
vJpG+wTml2aAjl4hy
66qtROx6nIJh1p0gZCGSc1FWX5t+X6/
nlXyZXuoUICRKYqxeU7Hve1Tmqmb2piY
8DmKxw==
=cP5X
-----END PGP PUBLIC KEY BLOCK-----
```

Add GPG key

Рис. 3.11: Вставка ключа на Github

Производим настройку автоматических подписей (рис. 3.12).

```
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global user.signingkey 33133502saq@gmail.com
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global commit.gpgsign true
[prihodko_ivan_228@prihodkoIvan ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 3.12: Настройка автоматических подписей

После этого авторизируемся на github с помощью gh, выбираем SSH протокол, публичный ключ id_rsa.pub и имя ключа sway (рис. 3.13).

```
[prihodko_ivan_228@prihodkoIvan ~]$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/prihodko_ivan_228/.ssh/id_rsa.pub
? Title for your SSH key: sway
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 3947-8D00
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/prihodko_ivan_228/.ssh/id_rsa.pub
✓ Logged in as SunHermit67
```

Рис. 3.13: Авторизация в gh

Создаем рабочую папку (рис. 3.14).

```
[prihodko_ivan_228@prihodkoIvan ~]$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
[prihodko_ivan_228@prihodkoIvan ~]$ cd ~/work/study/2024-2025/"Операционные системы"
[prihodko_ivan_228@prihodkoIvan Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository SunHermit67/study_2024-2025_os-intro on GitHub
https://github.com/SunHermit67/study_2024-2025_os-intro
```

Рис. 3.14: Создание рабочего пространства

Копируем туда репозиторий из лабораторной работы и создаем свой (рис. 3.15).

```
[prihodko_ivan_228@prihodkoIvan Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository SunHermit67/study_2024-2025_os-intro on GitHub
https://github.com/SunHermit67/study_2024-2025_os-intro
[prihodko_ivan_228@prihodkoIvan Операционные системы]$ git clone --recursive git@github.com:SunHermit67/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint:
```

Рис. 3.15: Создание репозитория

Удаляем ненужные файлы и создаем необходимые каталоги. Прописываем make prepare (рис. 3.16).

```
[prihodko_ivan_228@prihodkoIvan Операционные системы]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[prihodko_ivan_228@prihodkoIvan os-intro]$ rm package.json
[prihodko_ivan_228@prihodkoIvan os-intro]$ echo os-intro > COURSE
[prihodko_ivan_228@prihodkoIvan os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule      Update submules
[prihodko_ivan_228@prihodkoIvan os-intro]$ make prepare
```

Рис. 3.16: Удаление ненужных файлов и работа с рабочим пространством

Добавляем нашу папку для отправки, добавляем коммит и пушим! (рис. 3.17-3.19)

```
[prihodko_ivan_228@prihodkoIvan os-intro]$ git add .
```

Рис. 3.17: Добавление папки для отправки

```
[prihodko_ivan_228@prihodkoIvan os-intro]$ git commit -am 'feat(main): make course s
tructure'
```

Рис. 3.18: Добавление коммита

```
[prihodko_ivan_228@prihodkoIvan os-intro]$ git push
```

Рис. 3.19: Пуш

4 Выводы

Была произведена установка git, проведена его первоначальная настройка, были созданы ключи для авторизации и подписи, а также создан репозиторий курса из предложенного шаблона

5 Ответы на контрольные вопросы

1. Системы контроля версий – это системы, в которых мы можем хранить свои проекты и выкладывать их обновления, контролируя релизы и каждые внесённые изменения. Эти системы нужны для работы над проектами, чтобы иметь возможность контролировать версии проектов и в случае командной работы контролировать изменения, внесённые всеми участниками. Также, VCS позволяют откатываться на более ранние версии
2. Хранилище – репозиторий, в нём хранятся все файлы проекта и все его версии
commit – внесённые изменения в репозитории
история – это история изменений файлов проекта
рабочая копия – копия, сделанная из версии репозитория, с которой непосредственно работает сам разработчик
3. Централизованные системы контроля версий имеют один центральный репозиторий, с которым работают все разработчики. Примером является CVS, который является уже устаревшей системой.
В децентрализованных системах же используется множество репозиториях одного проекта у каждого из разработчиков, при этом репозитории можно объединять брать из каждого только то, что нужно. Примером является знакомый нам Git
4. Создаётся репозиторий, и разрабатывается проект. При внесении изменений файлы отправляются на сервер
5. Разработчик клонирует репозиторий к себе на компьютер, и после внесения

- изменений выгружает их на сервер в качестве отдельной версии. После этого разработчики с более высокими правами могут, например, объединить его версию с текущей
6. Хранение файлов проекта, а также обеспечение командной работы, и контроль за версиями проекта
 7. `git clone` – клонирует проект с сервера на компьютер
`git add` – добавляет папку для выгрузки на сервер
`git commit` – фиксирует изменения репозитория
`git push` – выгружает изменения на сервер
`git pull` – получить изменения с сервера
`git rm` – удалить файл
`git status` – получить статус репозитория
 8. С локальным: `git commit -am "added files"` – создаёт коммит С удалённым: `git push` – загрузить данные на удалённый сервер
 9. Ветки – это несколько независимых копий проекта, в каждой из которых ведётся разработка какой-то конкретной функции, при этом ветки существуют параллельно. Они нужны, когда нужно параллельно вести разработку нескольких функций, а в конце их можно объединить в одну
 10. Игнорировать файлы можно, внося их в файл `.gitignore`. Игнорировать файлы нужно, когда их не нужно добавлять в репозиторий. Например, это могут быть файлы виртуального окружения (`venv`)