

# **Отчет по лабораторной работе №2**

**Дисциплина: Операционные системы**

Иванов Сергей Владимирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>12</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

3.1	Установка git . . . . .	6
3.2	Установка gh . . . . .	6
3.3	Базовая настройка git . . . . .	6
3.4	Алгоритм rsa . . . . .	7
3.5	Алгоритм ed25519 . . . . .	7
3.6	Ключ pgr . . . . .	8
3.7	Копируем ключ . . . . .	8
3.8	Вставляем ключ . . . . .	9
3.9	Автоматические подписи . . . . .	9
3.10	Авторизация gh . . . . .	9
3.11	Создание репозитория . . . . .	10
3.12	Клонирование репозитория . . . . .	10
3.13	Удаление файла . . . . .	10
3.14	Создание каталогов . . . . .	10
3.15	git add . . . . .	11
3.16	Отправка на сервер . . . . .	11

# 1 Цель работы

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

## 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

### 3 Выполнение лабораторной работы

Установим git 'dnf install git'(рис. 1).

```
[root@svivanov1 ~]# dnf install git
Fedora 39 - x86_64 - Updates                               33 kB/s | 21 kB    00:00
Fedora 39 - x86_64 - Updates                               1.4 MB/s | 2.5 MB  00:01
Последняя проверка окончания срока действия метаданных: 0:00:06 назад, Сб 17 фев 2024 23:32:19.
Пакет git-2.43.1-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@svivanov1 ~]#
```

Рис. 3.1: Установка git

Установим gh 'dnf install gh' (рис. 2).

```
[root@svivanov1 ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:01:13 назад, Сб 17 фев 2024 23:32:19.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.43.1-1.fc39  updates      9.1 М
=====
Результат транзакции
=====
Установка: 1 Пакет
```

Рис. 3.2: Установка gh

Проведем базовую настройку git. Зададим имя и email владельца репозитория, настроим utf-8 в выводе сообщений git, зададим имя начальной ветки, укажем параметр autocrlf, параметр safecrlf (рис. 3).

```
[root@svivanov1 ~]# git config --global user.name "Сергей Иванов"
[root@svivanov1 ~]# git config --global user.email "1sergeiivanov1@mail.ru"
[root@svivanov1 ~]# git config --global init.defaultBranch master
[root@svivanov1 ~]# git config --global core.autocrlf input
[root@svivanov1 ~]# git config --global core.safecrlf warn
[root@svivanov1 ~]# git config --global core.quotepath false
[root@svivanov1 ~]#
```

Рис. 3.3: Базовая настройка git

Создадим ключ ssh по алгоритму rsa с ключём размером 4096 бит 'ssh-keygen -t rsa -b 4096' (рис. 4).

```
[root@svivanov1 ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8JkjbkoSro0Bbw0cKqiCWjTgbEmn8mw6fv/Fcoc9yzE root@svivanov1
The key's randomart image is:
+---[RSA 4096]-----+
|
|
|.. . .
|+. =   o o
|+0+. . S
|0=oo . ...o
|=oBo. o. = E
|B=o+.o + o =
|B=. o... o
+---[SHA256]-----+
```

Рис. 3.4: Алгоритм rsa

Создадим ключ ssh по алгоритму ed25519 'ssh-keygen -t ed25519' (рис. 5).

```
[root@svivanov1 ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:g+NkxRasDsBh8kHoy86yMQlBGXTU7ZMbFa4fQr/pbkc root@svivanov1
The key's randomart image is:
+--[ED25519 256]--+
|oB0+. ....
|o=+. . ooo
|o .. .o*.
| o . oOo
|o . o=oSo
|. + +.oo.+E
|* . +.
|. = .. .
|o. oo.
+---[SHA256]-----+
```

Рис. 3.5: Алгоритм ed25519

Создадим ключ `gpg` '`gpg --full-generate-key`', выбираем тип RSA and RSA, размер 4096, срок действия - не истекает никогда. (рис. 6).

```
[root@svivanov1 ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 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.

Выберите тип ключа:      I
  (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
```

Рис. 3.6: Ключ `gpg`

Пропускаем этап создания учетной записи GitHub, так как она уже создана

Выводим список ключей и копируем отпечаток приватного ключа '`gpg --list-secret-keys --keyid-format LONG`', скопируем наш сгенерированный PGP ключ в буфер обмена (рис. 7).

```
[root@svivanov1 ~]# 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, 0
[keyboard]
-----
sec   rsa4096/9B696B716EC35DED 2024-02-17 [SC]
      B2168BDE1A34FE648D086C559B696B716EC35DED
uid   [ абсолютно ] Sergey <1sergeiivanov1@mail.ru>
ssb   rsa4096/CF3CFD6BD4087343 2024-02-17 [E]

[root@svivanov1 ~]# gpg --armor --export <PGP Fingerprint> | xclip -sel clip
-bash: синтаксическая ошибка рядом с неожиданным маркером «|»
[root@svivanov1 ~]# gpg --armor --export 9B696B716EC35DED
```

Рис. 3.7: Копируем ключ

Перейдём в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода. (рис. 8).



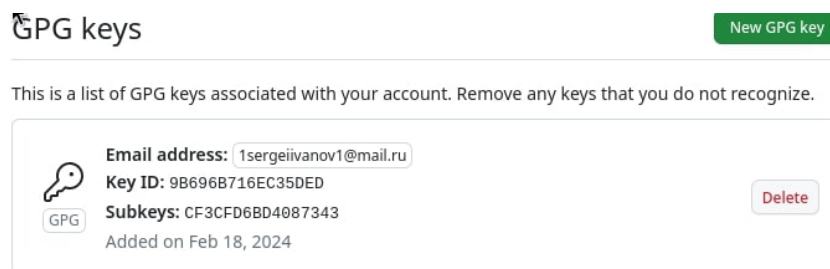


Рис. 3.8: Вставляем ключ

Настроим автоматические подписи коммитов git Используя введенный email, укажем Git применять его при подписи коммитов: (рис. 9).

```
[root@svivanov1 ~]# git config --global user.signingkey 9B696B716EC35DED
[root@svivanov1 ~]# git config --global commit.gpgsign tru
[root@svivanov1 ~]# git config --global commit.gpgsign true
[root@svivanov1 ~]# git config --global gpg.program $(which gpg2)
```

Рис. 3.9: Автоматические подписи

Авторизуемся для настройки gh через браузер (рис. 10).

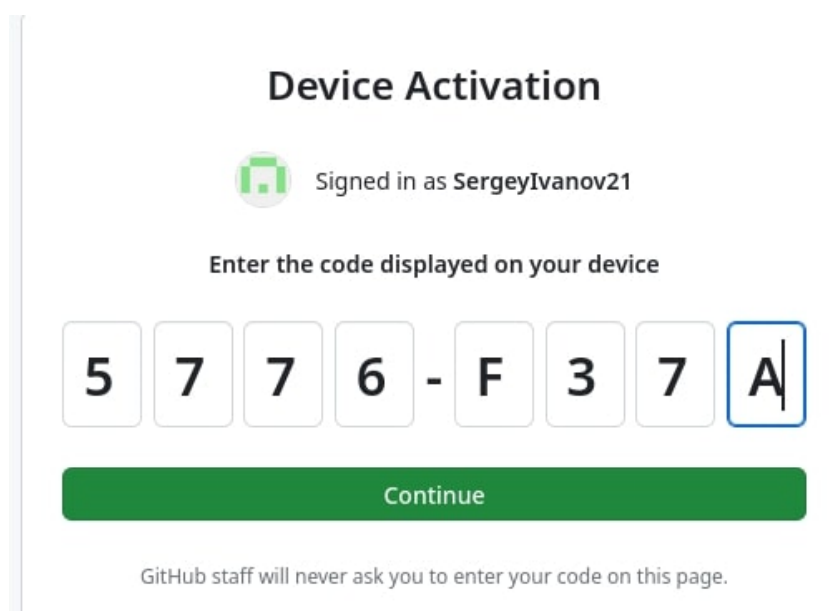


Рис. 3.10: Авторизация gh

Создадим репозиторий курса на основе шаблона и рабочее пространство (рис. 11).

```
[root@svivanov1 Операционные системы]# mkdir -p ~/work/study/2023-2024/"Операционные системы"
[root@svivanov1 Операционные системы]# cd
[root@svivanov1 ~]# mkdir -p ~/work/study/2023-2024/"Операционные системы"
[root@svivanov1 ~]# cd ~/work/study/2023-2024/"Операционные системы"
[root@svivanov1 Операционные системы]# gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository SergeyIvanov21/study_2023-2024_os-intro on GitHub
https://github.com/SergeyIvanov21/study_2023-2024_os-intro
[root@svivanov1 Операционные системы]#
```

Рис. 3.11: Создание репозитория

Клонируем репозиторий в рабочее пространство (рис. 12).

```
[root@svivanov1 Операционные системы]# git clone --recursive https://github.com/SergeyIvanov21/study_2023-2024_os-intro.git
Клонирование в «study_2023-2024_os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.60 КиБ | 388.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
```

Рис. 3.12: Клонирование репозитория

Перейдем в каталог курса и удалим лишние файлы (рис. 13).

```
[root@svivanov1 study_2023-2024_os-intro]# ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
[root@svivanov1 study_2023-2024_os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? y
[root@svivanov1 study_2023-2024_os-intro]# ls
CHANGELOG.md  COURSE  Makefile  README.git-flow.md  template
config        LICENSE  README.en.md  README.md
```

Рис. 3.13: Удаление файла

Создадим необходимые каталоги 'echo os-intro > COURSE' (рис. 14).

```
[root@svivanov1 os-intro]# echo os-intro > COURSE
[root@svivanov1 os-intro]# make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules

[root@svivanov1 os-intro]# make prepare
[root@svivanov1 os-intro]# ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md
config        labs    Makefile  presentation  README.en.md      README.md
[root@svivanov1 os-intro]#
```

Рис. 3.14: Создание каталогов

Отправим файлы на сервер 'git add' (рис. 15).

```
[root@svivanov1 os-intro]# git commit -am'feat(main): make course structure'
[master f8e42e0] feat(main): make course structure
361 files changed, 98413 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
```

Рис. 3.15: git add .

Добавляем комментарий и отправляем файлы (рис. 16).

```
[root@svivanov1 os-intro]# git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.14 КиБ | 21.38 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/SergeyIvanov21/study_2023-2024_os-intro.git
   048c7a1..f8e42e0  master -> master
[root@svivanov1 os-intro]#
```

Рис. 3.16: Отправка на сервер

## 4 Контрольные вопросы

**1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?** Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

**2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.** Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад») — синоним версии; процесс создания новой версии. 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. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.** Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.

**4. Опишите действия с VCS при единоличной работе с хранилищем.** В случае индивидуальной работы с хранилищем VCS пользователь создает, сохраняет и обновляет версии документов непосредственно в своем локальном хранилище.

**5. Опишите порядок работы с общим хранилищем VCS.** При работе с общим хранилищем VCS пользователи сначала загружают, обновляют и обмениваются версиями документов с помощью коммитов и пушей.

**6. Каковы основные задачи, решаемые инструментальным средством git?** У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

**7. Назовите и дайте краткую характеристику командам git.** `git --version` (Проверка версии Git) `git init` (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) `git clone --recursive https://www.github.com/username/repo-name` (Скопировать существующий удаленный Git-репозиторий) `git remote` (Просмотреть список текущих удалённых репозиториях Git) `git remote -v` (Для более подробного вывода) `git add my_script.py` (Можете указать в команде конкретный файл). `git add .` (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) `git commit -am "Commit message"` (Вы можете сжать все индексированные файлы и отправить коммит). `git branch` (Просмотреть список текущих веток можно с помощью команды `branch`) `git --help` (Чтобы узнать больше обо всех доступных параметрах и командах) `git push origin master` (Передать локальные коммиты в ветку удаленного репозитория).

**8. Приведите примеры использования при работе с локальным и удалённым репозиториями.** Пример использования локального репозитория: пользователь вносит изменения в код и сохраняет их локально. Пример использования удаленного репозитория: пользователь скачивает код с удаленного сервера, вносит изменения и отправляет их обратно на сервер.

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

ментов.

**10. Как и зачем можно игнорировать некоторые файлы при commit?** 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.

## 5 Выводы

В результате выполнения лабораторной работы мы изучили идеологию и применение средств контроля версий а также освоили умения по работе с git.