

Лабораторная работа №2

Отчёт

Борисенкова София Павловна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	19
5	Ответы на контрольные вопросы	20

Список иллюстраций

3.1	Установка git	7
3.2	Установка gh	8
3.3	Указание имени	8
3.4	Указание почты	9
3.5	Настройка кодировки utf8	9
3.6	Настройка git	9
3.7	Создание ключа RSA	10
3.8	Создание ключа ed22519	11
3.9	Создание ключа pgp (1)	12
3.10	Создание ключа pgp (2)	13
3.11	Список pgp ключей	14
3.12	Копирование ключа	14
3.13	Вставка ключа в GitHub	15
3.14	Настройка автоматических подписей коммитов git	15
3.15	Авторизация в gh	16
3.16	Создание рабочей директории и переход в неё	16
3.17	Создание репозитория курса	16
3.18	Клонирование репозитория	17
3.19	Удаление ненужных файлов и использование make	17
3.20	Использование git add	17
3.21	Использование git commit	18
3.22	Использование git push	18

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git (**tuis?**)

2 Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

Настроить подписи git.

Зарегистрироваться на Github.

Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

Для начала установим git. В моём случае он уже установлен (рис. 3.1)

```
[nsandryushin@nsandryushin ~]$ dnf install git
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на
  большинстве систем - под именем пользователя root).
[nsandryushin@nsandryushin ~]$ sudo dnf install git
[sudo] пароль для nsandryushin:
Последняя проверка окончания срока действия метаданных: 2:01:32 назад, П
т 23 фев 2024 19:21:01.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка git

Теперь установим gh (рис. 3.2)

```
[nsandryushin@nsandryushin ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 2:01:46 назад, П
т 23 фев 2024 19:21:01.
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

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]: у
Загрузка пакетов:
gh-2.43.1-1.fc39.x86_64.rpm          327 kB/s | 9.1 MB    00:28
-----
Общий размер          315 kB/s | 9.1 MB    00:29
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка           : 1/1
Установка            : gh-2.43.1-1.fc39.x86_64 1/1
Запуск скриптлета: gh-2.43.1-1.fc39.x86_64 1/1
Проверка             : gh-2.43.1-1.fc39.x86_64 1/1
Установлен:
gh-2.43.1-1.fc39.x86_64
Выполнено!
```

Рис. 3.2: Установка gh

Далее, зададим имя для владельца репозитория. В данном случае это моё имя (рис. 3.3)

```
[nsandryushin@nsandryushin ~]$ git config --global user.name "Nikita And
ryushin"
```

Рис. 3.3: Указание имени

Теперь зададим почту. Я задал почту, на которую у меня зарегистрирован аккаунт на github (рис. 3.4)


```
[nsandryushin@nsandryushin os-intro]$ git config --global user.email "me  
ga_nikitos111@mail.ru"
```

Рис. 3.4: Указание почты

Настроим кодировку utf8 в выводе сообщений git (рис. 3.5)

```
[nsandryushin@nsandryushin ~]$ git config --global core.quotepath false
```

Рис. 3.5: Настройка кодировки utf8

Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf (рис. 3.6)

```
[nsandryushin@nsandryushin ~]$ git config --global init.defaultBranch ma  
ster  
[nsandryushin@nsandryushin ~]$ git config --global core.autocrlf input  
[nsandryushin@nsandryushin ~]$ git config --global core.safecrlf warn
```

Рис. 3.6: Настройка git

Создадим ключ RSA размером 4096 бит (рис. 3.7)

```

[nsandryushin@nsandryushin ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nsandryushin/.ssh/id_rsa):
Created directory '/home/nsandryushin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nsandryushin/.ssh/id_rsa
Your public key has been saved in /home/nsandryushin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:RyDm2EExT0qlby0UPGcGc13TL0sb+Gh1ox4hr+LJ5cc nsandryushin@nsandryu
shin
The key's randomart image is:
+---[RSA 4096]---+
|      .0+B... .oo |
|      * 0o=+  . ..o|
|      . = o=.    o |
|      o o      . |
|      S o. +   |
|      . o. + =..|
|      . E *.+. |
|      o.+ +.+ |
|      .=.o.. |
+-----[SHA256]-----+

```

Рис. 3.7: Создание ключа RSA

Теперь создадим ключ по алгоритму ed22519 (рис. 3.8)

```

[nsandryushin@nsandryushin ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/nsandryushin/.ssh/id_ed25519)
:
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nsandryushin/.ssh/id_ed25519
Your public key has been saved in /home/nsandryushin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:JBI3EaM0R5Bt+qDzrx1Y0eS+oHSDriu02ch+bsADvQU nsandryushin@nsandryu
shin
The key's randomart image is:
+--[ED25519 256]--+
|  .+o Bo          |
|  E.oB o          |
|  ..+= + .        |
|  . +=. + o       |
| o.o=* . S        |
| o* .=.o .        |
| .o* . .          |
| +=. + .          |
| X++*+o           |
+-----[SHA256]-----+

```

Рис. 3.8: Создание ключа ed25519

Теперь создадим ключ gpg. Выбираем из предложенных вариантов первый тип (RSA and RSA), размер ключа задаём 4096 бит и делаем срок действия ключа неограниченным (рис. 3.9)

```

[nsandryushin@nsandryushin ~]$ 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.

gpg: создан каталог '/home/nsandryushin/.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

```

Рис. 3.9: ождение ключа pgp (1)

После нас попросят ввести свои данные. Мы вводим имя и адрес электронной почты. После этого соглашаемся с генерацией ключа (рис. 3.10)

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Nikita Andryushin
Адрес электронной почты: mega_nikitos111@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Nikita Andryushin <mega_nikitos111@mail.ru>"

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/nsandryushin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/nsandryushin/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/nsandryushin/.gnupg/openpgp-revocs.d/8C5B506CEDD386A3BFFC80D950243248E62B33F0.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-23 [SC]
      8C5B506CEDD386A3BFFC80D950243248E62B33F0
uid           Nikita Andryushin <mega_nikitos111@mail.ru>
sub   rsa4096 2024-02-23 [E]

```

Рис. 3.10: оздание ключа pgr (2)

Далее, выводим список pgr ключей (рис. 3.11)

```
[nsandryushin@nsandryushin ~]$ 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
[keyboxd]
-----
sec   rsa4096/50243248E62B33F0 2024-02-23 [SC]
      8C5B506CEDD386A3BFFC80D950243248E62B33F0
uid           [ абсолютно ] Nikita Andryushin <mega_nikitos111@mail
.ru>
ssb   rsa4096/7E067DD165DB2558 2024-02-23 [E]
```

Рис. 3.11: Список pgp ключей

Копируем наш ключ в буфер обмена (рис. 3.12)

```
[nsandryushin@nsandryushin ~]$ gpg --armor --export mega_nikitos111@mail
.ru | xclip -sel clip
```

Рис. 3.12: Копирование ключа

Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрал имя Sway (рис. 3.13)

Add new GPG key

Title

Sway

Key

```
vv
gv12xcIvwzgiZMbsDMAw7j17VIesK3zeVAKpLO8DhLLEKKJETG5tl+Hd
oFNHEaJL
cUtKmAmMx1tp0WhIqE2qUYarvPfmDTIC0rxI9wztezuoKjDRhM03xFc
2vrt/d1me
y56uwEMXb7g914WVcQSR7xBSYMz6jWICVws3Ku6UYLg6loBNR3Klo
pOg425I3ARt
2udMdEg72a1t4A==
=ZuG3
-----END PGP PUBLIC KEY BLOCK-----
```

Add GPG key

Рис. 3.13: Вставка ключа в GitHub

Теперь производим настройку автоматических подписей (рис. 3.14)

```
[nsandryushin@nsandryushin ~]$ git config --global user.signingkey mega_nikitos111@mail.ru
[nsandryushin@nsandryushin ~]$ git config --global commit.gpgsign true
[nsandryushin@nsandryushin ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 3.14: Настройка автоматических подписей коммитов git

После, нам нужно авторизоваться в github с помощью gh. Мы выбираем сайт для авторизации (GitHub.com), после выбираем предпочитаемый протокол (SSH), публичный SSH ключ (id_rsa.pub), и имя для ключа (Sway). В качестве способа

авторизации выбираем авторизацию через браузер (рис. 3.15)

```
[nsandryushin@nsandryushin ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/nsandryushin/.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: 1DE0-199A
Press Enter to open github.com 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/nsandryushin/.ssh/id_rsa.pub
✓ Logged in as DrNikiyProgrammingAccount
```

Рис. 3.15: Авторизация в gh

Теперь создаём рабочую директорию курса и переходим в неё (рис. 3.16)

```
[nsandryushin@nsandryushin ~]$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
[nsandryushin@nsandryushin ~]$ cd ~/work/study/2023-2024/"Операционные системы"
```

Рис. 3.16: Создание рабочей директории и переход в неё

Далее, создаём репозиторий для лабораторных работ из шаблона (рис. 3.17)

```
[nsandryushin@nsandryushin Операционные системы]$ gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository DrNikiyProgrammingAccount/study_2023-2024_os-intro on GitHub
https://github.com/DrNikiyProgrammingAccount/study_2023-2024_os-intro
```

Рис. 3.17: Создание репозитория курса

И клонируем его к себе на компьютер (рис. 3.18)


```
[nsandryushin@nsandryushin Операционные системы]$ git clone --recursive  
git@github.com:DrNikiyProgrammingAccount/study_2023-2024_os-intro.git os-  
intro  
Клонирование в «os-intro»...
```

Рис. 3.18: Клонирование репозитория

Переходим в него с помощью `cd` и удаляем ненужные файлы (`package.json`) и создаём необходимые каталоги, записав в файл `COURSE` строку `os-intro` (это наш текущий курс) и прописываем `make prepare` для того, чтобы нужные нам каталоги создались (рис. 3.19)

```
[nsandryushin@nsandryushin Операционные системы]$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro  
[nsandryushin@nsandryushin os-intro]$ rm package.json  
[nsandryushin@nsandryushin os-intro]$ echo os-intro > COURSE  
[nsandryushin@nsandryushin os-intro]$ make  
Usage:  
  make <target>  
  
Targets:  
  list           List of courses  
  prepare        Generate directories structure  
  submodule      Update submodules  
  
[nsandryushin@nsandryushin os-intro]$ make prepare
```

Рис. 3.19: Удаление ненужных файлов и использование `make`

Теперь добавляем нашу папку для отправки (рис. 3.20)

```
[nsandryushin@nsandryushin os-intro]$ git add .
```

Рис. 3.20: Использование `git add`

Делаем коммит, в котором указываем, что мы сделали структуру курса (рис. 3.21)

```
[nsandryushin@nsandryushin os-intro]$ git commit -am 'feat(main): make c
ourse structure'
[master aa04ea9] 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
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numer
ic.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__
.py
```

Рис. 3.21: Использование git commit

И отправляем файлы на сервер GitHub с помощью команды push (рис. 3.22)

```
[nsandryushin@nsandryushin 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), 342.12 КиБ | 1.41 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 github.com:DrNikiyProgrammingAccount/study_2023-2024_os-intro.git
733e347..aa04ea9 master -> master
```

Рис. 3.22: Использование git push

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 репозиториев 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`)