

Отчёта по лабораторной работе №2

Дисциплина: Операционные системы

Верниковская Екатерина Андреевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Установка программного обеспечения	7
Установка git	7
Установка gh	7
Базовая настройка git	8
Создание ключей ssh	9
Создание ключей pgr	10
Настройка github	12
Добавление PGP ключа в GitHub	13
Настройка автоматических подписей коммитов git	15
Настройка gh	15
Шаблон для рабочего пространства	18
Создание репозитория курса на основе шаблона	18
Настройка каталога курса	19
Контрольные вопросы + ответы	22
Выводы	27

Список таблиц

Список иллюстраций

1	Установка git	7
2	Установка gh	8
3	Задаём имя и email	8
4	Настройка utf-8	8
5	Имя начальной ветки	8
6	Параметры	9
7	Создание ключа ssh (1)	9
8	Создание ключа ssh (2)	10
9	Создание ключа pgr (1)	11
10	Пароль	11
11	Создание ключа pgr (2)	12
12	Аккаунт в github	12
13	Список ключей	13
14	Копирование PGP ключа (1)	13
15	Копирование PGP ключа (2)	14
16	Вставка полученного ключа в github	14
17	PGP ключ в github	15
18	Настройка автоматический подписей	15
19	Авторизация (1)	16
20	Авторизация (2)	16
21	Авторизация (3)	17
22	Авторизация (4)	18
23	Авторизация (5)	18
24	Создание репозитория (1)	18
25	Создание репозитория (2)	19
26	Переход в каталог курса	19
27	Удаление лишних файлов	19
28	Создание необходимых каталогов	20
29	Ввод команд	20
30	Ввод пароля	20
31	Работа команды git commit -am ‘...’	21
32	Отправка файлов	21

Цель работы

Изучение идеологии и применения средств контроля версий, а также освоение умения по работе с git.

Задание

1. Установить программное обеспечение.
2. Настроить git.
3. Создать ключи ssh.
4. Создать ключи pgr.
5. Настроить github.
6. Добавить PGP ключ в GitHub.
7. Настроить автоматические подписи коммитов git.
8. Настроить gh.
9. Создать репозиторий курса на основе шаблона.
10. Настроить каталог курса.

Выполнение лабораторной работы

Установка программного обеспечения

Установка git

Устанавливаем git, введя `dnf install git` (рис. [-@fig:001])

```
[eavernikovskaya@eavernikovskaya ~]$ sudo dnf install git
[sudo] пароль для eavernikovskaya:
Попробуйте ещё раз.
[sudo] пароль для eavernikovskaya:
Fedora 39 - x86_64 - Updates 19 kB/s | 28 kB 00:01
Fedora 39 - x86_64 - Updates 1.9 MB/s | 4.6 MB 00:02
Последняя проверка окончания срока действия метаданных: 0:00:19 назад, Чт 22 фев 2024 16:44:14.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 1: Установка git

Установка gh

Устанавливаем gh, введя `dnf install gh` (рис. [-@fig:002])

```
eavernikovskaya@eavernikovskaya ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:01:28 назад, Чт 22 фев 2024 16:44:14.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Резепозиторий  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]: y
Загрузка пакетов:
gh-2.43.1-1.fc39.x86_64.rpm 4.8 МВ/с | 9.1 МВ 00:01
-----
Общий размер 3.9 МВ/с | 9.1 МВ 00:02
Проверка транзакции
Проверка транзакции успешно завершена.
Дает проверка транзакции
Чест транзакций проведен успешно.
Выполнение транзакции
Подготовка :
Установка : gh-2.43.1-1.fc39.x86_64 1/1
Запуск скрипта: 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!
eavernikovskaya@eavernikovskaya ~]$
```

Рис. 2: Установка gh

Базовая настройка git

Задаём имя и email владельца репозитория (рис. [-@fig:003])

```
[eavernikovskaya@eavernikovskaya ~]$ git config --global user.name "Katerok27153"
[eavernikovskaya@eavernikovskaya ~]$ git config --global user.email "1132236136@pfur.ru"
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 3: Задаём имя и email

Настраиваем utf-8 в выводе сообщений git с помощью git config --global core.quotePath false (рис. [-@fig:004])

```
[eavernikovskaya@eavernikovskaya ~]$ git config --global core.quotePath false
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 4: Настройка utf-8

Задаём имя начальной ветки (будем называть её master). Для этого мы вводим команду git config --global init.defaultBranch master (рис. [-@fig:005])

```
[eavernikovskaya@eavernikovskaya ~]$ git config --global init.defaultBranch master
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 5: Имя начальной ветки

Вводим параметры autocrlf и safecrlf рис. [-@fig:006])

```
[eavernikovskaya@eavernikovskaya ~]$ git config --global core.autocrlf input
[eavernikovskaya@eavernikovskaya ~]$ git config --global core.safecrlf warn
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 6: Параметры

Создание ключей ssh

Создаём ключ ssh по алгоритму rsa с ключём размером 4096 бит, введя команду ssh-keygen -t rsa -b 4096 (рис. [-@fig:007])

```
[eavernikovskaya@eavernikovskaya ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/eavernikovskaya/.ssh/id_rsa):
Created directory '/home/eavernikovskaya/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/eavernikovskaya/.ssh/id_rsa
Your public key has been saved in /home/eavernikovskaya/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8HPWdIWtGYCL+SIVCdPDqc6ThifjRVGqG8CuMr72d5s eavernikovskaya@eavernikovskaya
The key's randomart image is:
+---[RSA 4096]---+
|      o=.o ... o. |
|      .oB .   o.. |
|      o  oo = .. .+ |
|      . .oo+ .o .o |
|      . o= oS.o .   |
|      . +oX .+.    |
|+      ..* o .     |
|oo      .. ..      |
|.oo..   .E.        |
+---[SHA256]-----+
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 7: Создание ключа ssh (1)

Далее создаём ключ ssh по алгоритму ed25519, введя ssh-keygen -t ed25519 (рис. [-@fig:008])

```

[eavernikovskaya@eavernikovskaya ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/eavernikovskaya/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/eavernikovskaya/.ssh/id_ed25519
Your public key has been saved in /home/eavernikovskaya/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:eJsIxAVwvSreSaiQ5MNP+5Dk4Pacsa7IW9jmEPZpNU eavernikovskaya@eavernikovskaya
The key's randomart image is:
+--[ED25519 256]--+
|.++..|
|..+|
|...=|
|oo+* E |
|+o&.o . S|
|o&o= . o o|
|*=Boo . o|
|*+*+|
|o*..|
+----[SHA256]-----+
[eavernikovskaya@eavernikovskaya ~]$ █

```

Рис. 8: Создание ключа ssh (2)

Создание ключей pgr

Генерируем ключ, с помощью `gpg --full-generate-key` и указываем нужные данные при создании (рис. [-@fig:009])

```

[eavernikovskaya@eavernikovskaya ~]$ 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/eavernikovskaya/.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. 9: Создание ключа gpg (1)

При создании ключа, у нас потребуют придумать пароль. Вводим пароль и всё готово! (рис. [-@fig:010]), (рис. [-@fig:011])

[3066]@eavernikovskaya (gpg --full-generate-k

 **Фраза-пароль:**
Введите фразу-пароль
для защиты нового ключа

Пароль:

Подтверждение:

Отмена (C) ОК

Рис. 10: Пароль

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: VernikovskayaEkaterina
Адрес электронной почты: 1132236136@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "VernikovskayaEkaterina <1132236136@pfur.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
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/eavevnikovskaya/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/eavevnikovskaya/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/eavevnikovskaya/.gnupg/openpgp-revocs.d/
001AB0F571DA3173CB2CCCA647519BDC62A55FE7.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-22 [SC]
       001AB0F571DA3173CB2CCCA647519BDC62A55FE7
uid           VernikovskayaEkaterina <1132236136@pfur.ru>
sub   rsa4096 2024-02-22 [E]

```

Рис. 11: Создание ключа gpg (2)

Настройка github

В прошлом семестре я уже создала аккаунт github. Поэтому мне ничего настраи-
вать не нужно (рис. [-@fig:012])

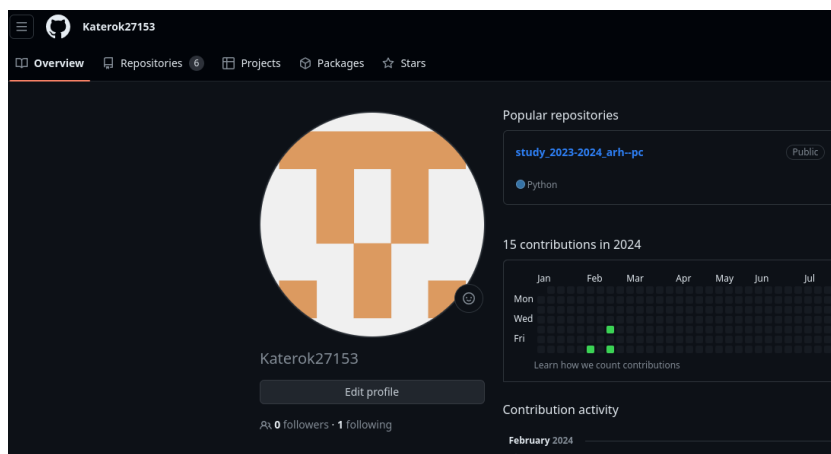


Рис. 12: Аккаунт в github

Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа, введя команду `gpg --list-secret-keys --keyid-format LONG` (рис. [-@fig:013])

```
[eavernikovskaya@eavernikovskaya ~]$ 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, 0f, 1u
[keyboard]
-----
sec   rsa4096/47519BDC62A55FE7 2024-02-22 [SC]
      001AB0F571DA3173CB2CCCA647519BDC62A55FE7
uid   [ абсолютно ] VernikovskayaEkaterina <1132236136@pfur.ru>
ssb   rsa4096/69A1072281DCADFE 2024-02-22 [E]

[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 13: Список ключей

Копируем отпечаток ключа.

Далее копируем наш сгенерированный PGP ключ в буфер обмена, с помощью команды `gpg --armor --export | xclip -sel clip` (рис. [-@fig:014])

```
[eavernikovskaya@eavernikovskaya ~]$ gpg --armor --export 47519BDC62A55FE7 | xclip -sel clip
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 14: Копирование PGP ключа (1)

У меня не сработала эта команда, поэтому я с помощью команды `pg --armor --export` посмотрела мой сгенерированный ключ и скопировала его ручками (рис. [-@fig:015])

```
[eavernikovskaya@eavernikovskaya ~]$ gpg --armor --export 47519BDC62A55FE7
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXXZ0sBEACo2jGJQVY4tEBTuW5IazhXVGdHsk0TjE4oQxE3WAaIUcgL3HbK
ZRKFccSjjgI06A+QIVcSJVfAsfqHjTJ59V9gX8pbvAvdKiKuUpDyHEMPB+LxL4P
KcqM7mDH+7gpkLd3ivgUBjMPXFWmW3a0ZtnTm/qsbvcUvd3yfkMrEs1p5XSKxo1M
20GP7e90JbY4xGjg/La67j/7bEvFnir2gTU+au/FtZjm+qnVmcuK5XQQR2bFkI1H
mQwgFANK1AStFqiQpdStUcuCE8cqy8y2LvcrJPaQ/hGECa++10txk0Za9CXws1e6
K21wznK2y+0If9IvG3CJkimbFpMZHesL5tEMERDo81ApqsiG/qic3cCQW14XntS
F14CkQJh0nclM4g8Xx8WJFFRcTukar1pNisIn15Cu/Dw3H5qS2D4WAxTpQ0h2QnL
PI2ZL2VYAP9vzbnPZ20BJrPr3a1Ev90rm5PEPis1Is+7E/IIgeXx06jRKq5ZGje7
Nv0G94a48/1eJTJeWt0JBdYgdENaqXQhNFwwW9zqWL+dEBK20ft24XT/7gW1K7qq
cD4SYpSd5dtUN6xwyIaUdKFO0uPG7i1w518g032BSUX8p88UtinEKAf307Ve3xdI
7nELY6kGSHu2ID4GhxZtT0JsC4J3hopLwDE12JGqCN1H7B9Yn0h/AzTqHwARAQAB
tCtWZXJuaWtvdnNrYX1hRWthdGVyaW5hIDwxMTMyMjM2MTM2QHBmdXIucnU+1QJR
BBMBCAA7FiEEABqw9XHaMXPLLMymR1Gb3GK1X+cFAmXXZ0sCGwMFCwkIBwICIGIG
FQoJCAAsCBByCAwECHgcCF4AACgkQR1Gb3GK1X+eQLA/7BsBJAa+o5+He8AYZPg9j
```

Рис. 15: Копирование PGP ключа (2)

Далее переходим в настройки GitHub (<https://github.com/settings/keys>), нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода (рис. [-@fig:016]), (рис. [-@fig:017])

Add new GPG key

Title

Key

```
7J0H4p+C
ntGeCwNzQcQNSH1IRhhzjY4Q2PZ+AMLQ0rGCVSaoq3zbVNpwK+Vk
qz436X0li/im
S2yN+TGieX/
lQZDefQV5AJoAXCmm6kXfN+k4TuXww+sNc0wO1P3I3ZqRI/VWJYD5
ZD/NA5czw4sYlg==
=EQ48
-----END PGP PUBLIC KEY BLOCK-----
```


Add GPG key

Рис. 16: Вставка полученного ключа в github

GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.


GPG

Email address: 1132236136@pfur.ru
Key ID: 47519BDC62A55FE7
Subkeys: 69A1072281DCADFE
Added on Feb 22, 2024

[Delete](#)

Рис. 17: PGP ключ в github

Настройка автоматических подписей коммитов git

Используя введённый email, указываем Git применять его при подписи коммитов (рис. [-@fig:018])

```
[eavernikovskaya@eavernikovskaya ~]$ git config --global user.signingkey 47519BDC62A55FE7
[eavernikovskaya@eavernikovskaya ~]$ git config --global commit.gpgsign true
[eavernikovskaya@eavernikovskaya ~]$ git config --global gpg.program $(which gpg2)
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 18: Настройка автоматический подписей

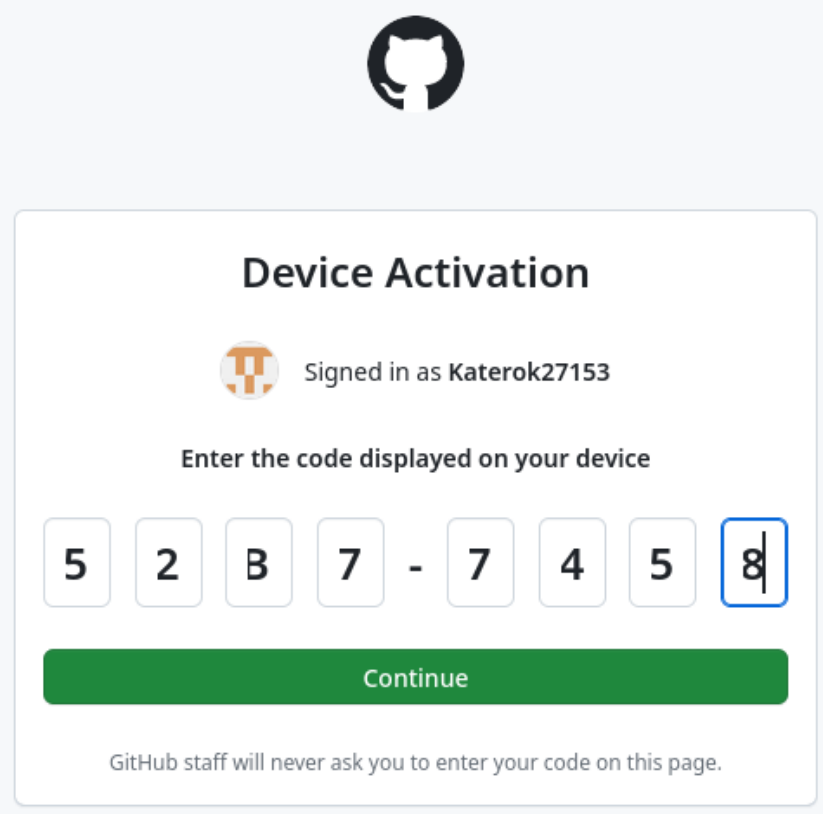
Настройка gh

Для начала нам необходимо авторизоваться. Для этого вводим команду `gh auth login`. Мы авторизуемся через браузер (рис. [-@fig:019]), (рис. [-@fig:020]), (рис. [-@fig:021]), (рис. [-@fig:022]), (рис. [-@fig:023])

```
[eavernikovskaya@eavernikovskaya ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser


! First copy your one-time code: 5287-7458
Press Enter to open github.com in your browser...
█
```

Рис. 19: Авторизация (1)



The image shows a GitHub Device Activation screen. At the top is the GitHub logo. Below it, the title "Device Activation" is centered. Under the title is a small orange icon and the text "Signed in as Katerok27153". Below this, the instruction "Enter the code displayed on your device" is centered. The code "5287-7458" is displayed in a row of eight boxes, with the last box containing a cursor. Below the code boxes is a green "Continue" button. At the bottom, a small note states "GitHub staff will never ask you to enter your code on this page."

Device Activation

 Signed in as Katerok27153

Enter the code displayed on your device

5 2 8 7 - 7 4 5 8

Continue

GitHub staff will never ask you to enter your code on this page.

Рис. 20: Авторизация (2)

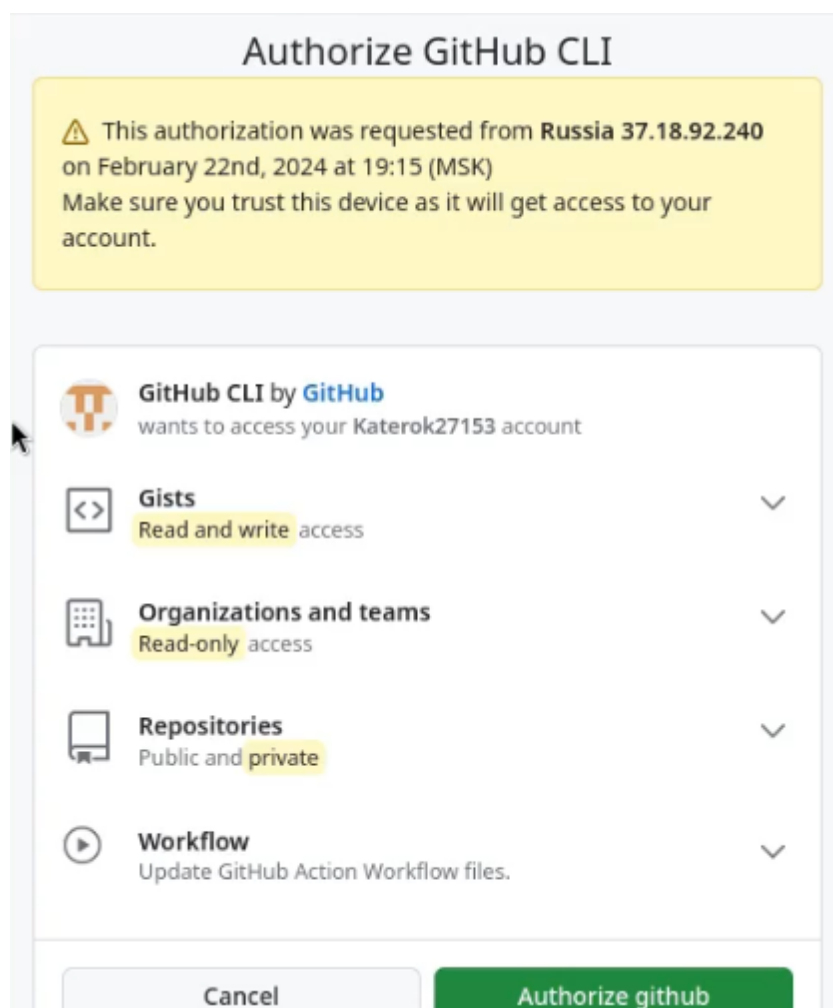


Рис. 21: Авторизация (3)

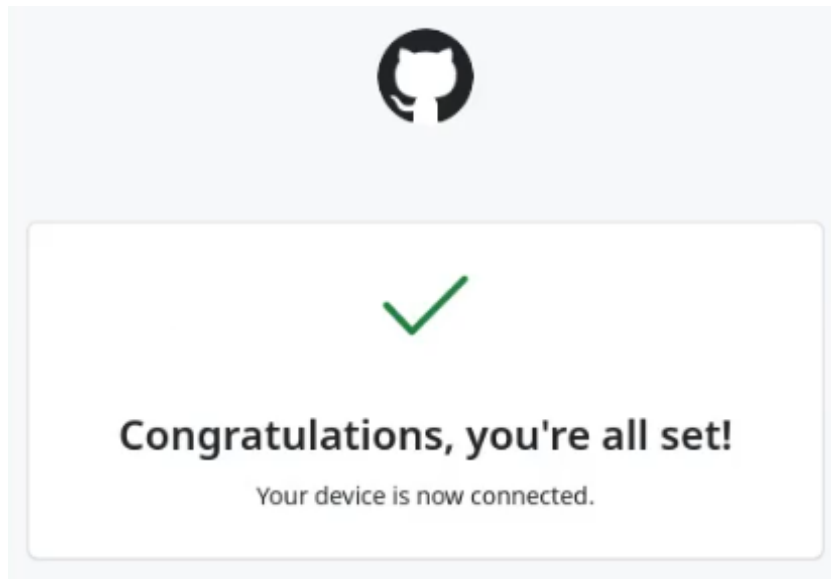


Рис. 22: Авторизация (4)

```
[eavernikovskaya@eavernikovskaya ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 52B7-7458
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as Katerok27153
```

Рис. 23: Авторизация (5)

Шаблон для рабочего пространства

Создание репозитория курса на основе шаблона

Создаём репозиторий курса на основе шаблона (рис. [-@fig:024]), (рис. [-@fig:025])

```
[eavernikovskaya@eavernikovskaya ~]$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
[eavernikovskaya@eavernikovskaya Операционные системы]$ gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository Katerok27153/study_2023-2024_os-intro on GitHub
https://github.com/Katerok27153/study_2023-2024_os-intro
[eavernikovskaya@eavernikovskaya Операционные системы]$
```

Рис. 24: Создание репозитория (1)

```
[eavernikovskaya@eavernikovskaya Операционные системы]$ git clone --recursive https://github.com/Katerok27153/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 KiB | 405.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/eavernikovskaya/work/study/2023-2024/Операционные системы/study_2023-2024_os-intro/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 KiB | 168.00 KiB/c, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/eavernikovskaya/work/study/2023-2024/Операционные системы/study_2023-2024_os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 KiB | 1.88 MiB/c, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c68a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8c8b2d67caeb8a19ef028ced88e'
```

Рис. 25: Создание репозитория (2)

Настройка каталога курса

Переходим в каталог курса (рис. [-@fig:026])

```
[eavernikovskaya@eavernikovskaya Операционные системы]$ cd ~/work/study/2023-2024/Операционные системы/os-intro
[eavernikovskaya@eavernikovskaya os-intro]$
```

Рис. 26: Переход в каталог курса

Удаляем все лишние файлы, введя `rm package.json` (рис. [-@fig:027])

```
[eavernikovskaya@eavernikovskaya os-intro]$ rm package.json
[eavernikovskaya@eavernikovskaya os-intro]$
```

Рис. 27: Удаление лишних файлов

Создаём необходимые каталоги (рис. [-@fig:028])

```

[eaavernikovskaya@eaavernikovskaya os-intro]$ echo os-intro > COURSE
[eaavernikovskaya@eaavernikovskaya os-intro]$ make list
bash: make: команда не найдена
[eaavernikovskaya@eaavernikovskaya os-intro]$ make list
net-admin    Администрирование локальных сетей
net-os-admin  Администрирование сетевых подсистем
arch-pc       Архитектура ЭВМ
sciprog-intro Введение в научное программирование
infosec       Информационная безопасность
computer-practice Компьютерный практикум по статистическому анализу данных
mathsec       Математические основы защиты информации и информационной безопасности
mathmod       Математическое моделирование
simulation-networks Моделирование сетей передачи данных
sciprog        Научное программирование
os-intro      Операционные системы
[eaavernikovskaya@eaavernikovskaya os-intro]$ make prepare
[eaavernikovskaya@eaavernikovskaya os-intro]$ make submodule
git submodule update --init --recursive
git submodule foreach 'git fetch origin; git checkout $(git rev-parse --abbrev-ref HEAD); git reset --hard origin/$(git rev-parse --abbrev-ref HEAD); git submodule update --recursive; git clean -dfx'
Entering 'template/presentation'
Указатель HEAD сейчас на коммите 40a1761 Merge branch 'release/1.0.3'
Entering 'template/report'
Указатель HEAD сейчас на коммите 7c31ab8 Merge branch 'release/1.0.4'
[eaavernikovskaya@eaavernikovskaya os-intro]$

```

Рис. 28: Создание необходимых каталогов

Далее отправляем файлы на сервер. Вводим команды `git add .` и `git commit -am 'feat(main): make course structure'`. После этого вводим наш пароль (рис. [fig:029]), (рис. [fig:030]), (рис. [fig:031])

```

[eaavernikovskaya@eaavernikovskaya os-intro]$ git add .
[eaavernikovskaya@eaavernikovskaya os-intro]$ git commit -am 'feat(main): make course structure'

```

Рис. 29: Ввод команд

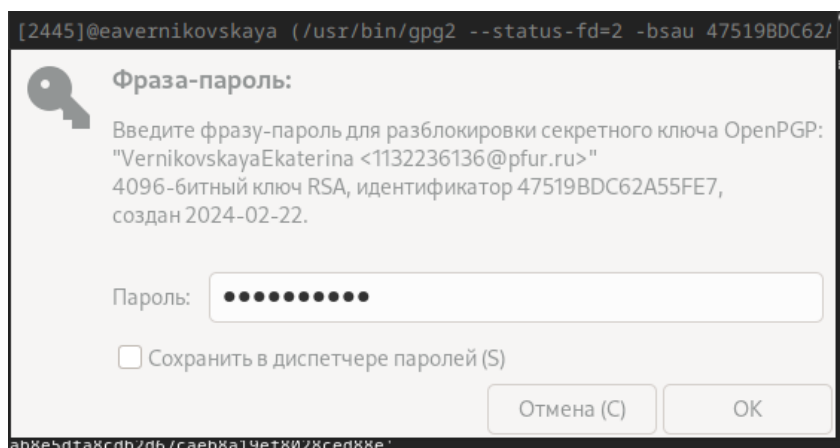


Рис. 30: Ввод пароля

```

create mode 100644 project-personal/stage4/report/image/placemg_800_600_tech.jpg
create mode 100644 project-personal/stage4/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage4/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage4/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage4/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage4/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage4/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage4/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage4/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage4/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage4/report/report.md
create mode 100644 project-personal/stage5/presentation/Makefile
create mode 100644 project-personal/stage5/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage5/presentation/presentation.md
create mode 100644 project-personal/stage5/report/Makefile
create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placemg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placemg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md

```

Рис. 31: Работа команды git commit -am ‘...’

Отправляем файлы на сервер, введя git push (рис. [-@fig:032])

```

[eavernikovskaya@eavernikovskaya os-intro]$ git push
Перечисление объектов: 39, готово.
Подсчет объектов: 100% (39/39), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.07 Киб | 9.25 Миб/с, готово.
Всего 38 (изменений 4), повторно использовано 1 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/Katerok27153/study_2023-2024_os-intro.git
   fd27d52..769626b  master -> master
[eavernikovskaya@eavernikovskaya os-intro]$

```

Рис. 32: Отправка файлов

Контрольные вопросы + ответы

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

Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.

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

хранилище - это репозиторий, в котором хранятся все файлы и документы, включая историю изменений. commit - отслеживание и сохранение изменений. история - сохраняет в себе изменения проекта на всех этапах. рабочая копия - копия проекта, связанная с репозиторием.

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

Централизованные VCS: В централизованных VCS весь код и его история хранятся в одном центральном репозитории. Разработчики работают с копией основного репозитория на своих локальных машинах, откуда отправляют изменения в центральное хранилище.

Примеры: - Subversion (SVN): Один из популярных централизованных VCS. Разработчики могут коммитить изменения в центральный репозиторий и обновлять

свои локальные копии. - Perforce: Еще один пример системы контроля версий с централизованным подходом, который широко применяется в больших коммерческих проектах.

Децентрализованные VCS: Децентрализованные VCS позволяют каждому участнику проекта иметь полноценную копию всего репозитория. Это означает, что разработчики имеют доступ ко всей истории проекта локально и могут работать независимо от подключения к сети.

Примеры: - Git: Самая популярная децентрализованная система контроля версий. Разработчики могут коммитить, откатывать изменения и создавать ветки без необходимости доступа к центральному серверу. - Mercurial: Еще один пример децентрализованной VCS, обеспечивающий высокую скорость работы и гибкость в управлении кодом.

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

При единоличной работе с хранилищем (репозиторием) в системе контроля версий (VCS), разработчик ведет работу над кодом самостоятельно без коллективного взаимодействия. В такой ситуации основной упор делается на сохранение версий кода и отслеживание изменений для личного удобства и безопасности. Вот основные действия, которые могут выполняться при единоличной работе с хранилищем:

- Инициализация репозитория
- Клонирование репозитория
- Коммит изменений
- Просмотр истории изменений
- Создание веток
- Обновление репозитория
- Удаление, перемещение файлов
- Откат изменений
- Игнорирование файлов
- Резервное копирование

5. Опишите порядок работы с общим хранилищем VCS.

Работа с общим хранилищем (репозиторием) в системе контроля версий (VCS) включает в себя совместную работу нескольких разработчиков над одним проектом. Вот порядок действий при работе с общим хранилищем VCS:

- Создание или клонирование репозитория
- Получение последних изменений
- Внесение изменений
- Коммит
- Отправка изменений (push)
- Работа с веткам и т.д.

6. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи, решаемые инструментальным средством Git, включают в себя управление версиями кода, обеспечение совместной работы над проектами, отслеживание изменений, создание и объединение ветвей разработки, а также возможность отката к предыдущим версиям кода. Git также предоставляет возможность создания резервных копий (backup) и управление изменениями в коде, что делает его ключевым инструментом для разработчиков программного обеспечения.

7. Назовите и дайте краткую характеристику командам git.

- git init - создание основного дерева репозитория
- git pull - получение обновлений (изменений) текущего дерева из центрального репозитория
- git push - отправка всех произведённых изменений локального дерева в центральный репозиторий
- git status - просмотр списка изменённых файлов в текущей директории
- git diff - просмотр текущих изменений
- git add . - добавить все изменённые и/или созданные файлы и/или каталоги

- `git add имена_файлов` - добавить конкретные изменённые и/или созданные файлы и/или каталоги
- `git rm имена_файлов` - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
- `git commit -am 'Описание коммита'` - сохранить все добавленные изменения и все изменённые файлы
- `git commit` - сохранить добавленные изменения с внесением комментария через встроенный редактор
- `git checkout -b имя_ветки` - создание новой ветки, базирующейся на текущей
- `git checkout имя_ветки` - переключение на некоторую ветку
- `git push origin имя_ветки` - отправка изменений конкретной ветки в центральный репозиторий
- `git merge --no-ff имя_ветки` - слияние ветки с текущим деревом
- `git branch -d имя_ветки` - удаление локальной уже слитой с основным деревом ветки
- `git branch -D имя_ветки` - принудительное удаление локальной ветки
- `git push origin :имя_ветки` - удаление ветки с центрального репозитория

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

Git pull: Команда `git pull` используется для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым. Слияние удаленных вышестоящих изменений в локальный репозиторий — это обычное дело в процессе совместной работы на основе Git.

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

Ветви (branches) в контексте систем контроля версий, таких как Git, представляют собой параллельные линии разработки, которые позволяют команде разработчиков работать над отдельными фрагментами кода независимо друг от друга. Ветви могут

быть полезны для разработки новых функций, исправления ошибок, экспериментов с кодом и поддержания стабильной основной версии программного обеспечения. Создание и использование ветвей помогает упростить процесс разработки, избегая конфликтов и обеспечивая возможность параллельной работы над различными задачами.

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

Игнорируемый файл — файл, явным образом помеченный для Git как файл, который необходимо игнорировать.

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. Вот некоторые распространенные примеры таких файлов (например: `/bin`, `.lock`, `.tmp`, `/packages`)

Игнорируемые файлы отслеживаются в специальном файле `.gitignore`, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл `.gitignore`, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы `.gitignore` содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.

Выводы

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