

Отчёт по лабораторной работе №2

Специальность: архитектура компьютеров

Курушин Георгий Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Установка git и gh	9
4.2	Базовая настройка git.	9
4.3	Создание PGP ключа.	10
4.3.1	Добавление ключа на ГитХаб.	12
4.3.2	Настройка автоматических подписей коммитов git	13
4.4	Настройка gh	14
4.5	Создание и настройка репозитория курса.	15
5	Контрольные вопросы	18
6	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Ввод команд в терминал	10
4.2	Создание нового pg ключа	11
4.3	Вывод ключа в терминал	12
4.4	Новый ключ PGP	13
4.5	Настройка необходимых подписей коммитов	14
4.6	Настройка gh и авторизация в браузере	15
4.7	Созданный репозиторий, папка первой лабораторной работы . . .	16
4.8	Отправка файлов на сервер	17

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение 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 Теоретическое введение

Системы контроля версий (Version Control System, 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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

4 Выполнение лабораторной работы

4.1 Установка git и gh

Установим гит командой `dnf install git`, установим gh командой `dnf install gh`

4.2 Базовая настройка git.

Открываем терминал. При помощи команд `git config --global user.name` и `git config --global user.email` зададим имя пользователя и адрес электронной почты. При помощи команды `git config --global core.quotePath false` настроим utf-8 в выводе сообщений git. (рис. 4.1)

```
foot
grkurushin@grkurushin:~$ dnf install git
Для выполнения запрошенной операции требуются привилегии суперпользователя. Пожалуйста, войдите
в систему как пользователь с повышенными правами или используйте опции "--assumeno" или "--down
ploadonly", чтобы выполнить команду без изменения состояния системы.
grkurushin@grkurushin:~$ sudo dnf install git
[sudo] пароль для grkurushin:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
grkurushin@grkurushin:~$ 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% | 10.3 MiB/s | 10.3 MiB | 00m01s
-----
[1/1] Total                          100% | 5.7 MiB/s | 10.3 MiB | 00m02s
Выполнение транзакции
[1/3] Проверить файлы пакета          100% | 30.0 B/s | 1.0 B | 00m00s 00m00s
[2/3] Подготовить транзакцию           100% | 2.0 B/s | 1.0 B | 00m00s 00m00s
[3/3] Установка gh-0:2.65.0-1.fc41.x86_64 100% | 26.7 MiB/s | 42.7 MiB | 00m02s
Завершено!
grkurushin@grkurushin:~$ git config --global user.name "Georgy Kurushin"
grkurushin@grkurushin:~$ git config --global user.email "1132246755@pfur.ru"
grkurushin@grkurushin:~$ git config --global core.quotepath false
grkurushin@grkurushin:~$
```

Рис. 4.1: Ввод команд в терминал

4.3 Создание PGP ключа.

Генерируем ключ командой `gpg --full-generate-key`, настраиваем его по заданным требованиям. (рис. 4.2)

```

foot
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)
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Georgy
Адрес электронной почты: 1132246755@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Georgy <1132246755@pfur.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.
^[[B^[[B^[[BНеобходимо 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/grkurushin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/grkurushin/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/grkurushin/.gnupg/openpgp-revocs.d/4AA51FB0E2AF43841FBD
829C638D4A994CF6B8C7.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2025-03-04 [SC]
       4AA51FB0E2AF43841FBD829C638D4A994CF6B8C7
uid           Georgy <1132246755@pfur.ru>
sub   rsa4096 2025-03-04 [E]

grkurushin@grkurushin: $ 
[2] 0:gpg*                                     "grkurushin" 23:07 04-мар-25

```

Рис. 4.2: Создание нового pg ключа

Выводим ключ в терминал командой `gpg --list-secret-keys --keyid-format LONG`.
После этого экспортируем его командой `gpg --armor --export`. (рис. 4.3)

```
foot
tBtH2W9yZ3kgPDExMzIyNDY3NTVAcGZ1ci5ydT6JAlEEwEiADsWIQRKpR+w4q9D
hB+9gpxjjUqZTPa7xwUCZ8ddjAibAwULCQgHAgIiAgYVCgICwIEFgIDAQIeBwIX
gAAKCRBjjUqZTPa7x2z+D/9ocgSjpuUOK0BqE05ChRR8HKvU4L1rWwnmwOpVjxsg
lBcOcsPzvztHL35CugJDOEt1aEwXT/n5461pxE4k761v1JqgatzY/jjqb7R7AlYA
jdcnzczRUQlwM3m2aBn6wbMGZdymWg1x5Gkuph7tMTu5h0g/rkseAN3jOK/qEe8zJ
TpWg1EuZdrTmm1GEzs7J+u7IyQkvCrTzrLnOt1ckmUvFuzLGF2eR0W0nog0gT8j9
HYYGXewwiTCC0ij40kcam0t7H1rBxeA6n55V6AaiW5dIRk1wX9HTT4KH4j5vFMh
REPK1ndAeo145Axbhyr4pLJmKDcvSJEMe4ru405p05dtnxP6dhdF15BQZ/2MDdE
CaWExDQcWyaZGE50E3qa+YmnX5/Emoi0aZIXm07gUjsEZbtuw5ZMGU6vvyicNn1Hp
CH9dK5mISJ1kXFn/z+AxYjb78uZKCVrKxy8SYrPt1kMxX5jnpZ0eV1MwtNhaTBRR
bm571isGNVaGeA8ms1A9JAKrFB/iae89r6tRy5tRaR3p+uz0e0aYKQt5+fjiP3HP
qCH/6gx2r0+cXhusV/RLzWtrVWJQE+KMu1htwncBg6dd2QSMbIt5R3Q21s0M9od
rC0GczF00KeU8WzIEGfbJtpHuT2C5U7kmjjqia0JW7zsGfBMDaHYJLj1eas9M40
RrKCDQRnx12MARAAyn9uMHG1jVogmAmk9HBTQx0PwC/6QYVKeVcPl6bI6fbYc0RT
8WeFocTa1Ba7Uuuu41Yv2r5RtA26QcbOvFnVn1BmVal6ShnKoQTounoHwZWJmkoy
tv9063EC6k4yk1E7IuTeGN91ea/WE+jvwSWwHjZU0hHRrHKJgvoc2S0MJL1l3/Z
j2Ch2NwKOC3+XH1eZemNCS4JRM1truCRYWxiq0h2bUjb0Q8GQ4FLngp9+788GBI
wa9pb06hhgbN842Vmqg4UQHyzvrgtkySGTeDGTnmX+b/4XBTTipNk+K8tzWTZ4
0LUM0ZLISPvTECurtGXnh0j/9uu666G7K8oLnuhFRHYzrFk7Yya8X2M7p0obb03
uwVUbvRqwoE4cSTPpg/IUz35Jxe1yQBWTsxXj0WxrA8pB8YzqOCjSAK1vi9QdydY
lXJjaSSThake1Dowyc+UbHACxWFnH2Gsba2D3jjMg3QK6t436A004lkayb/RXum3
ShKJTYRs+T990dAJf114tDcvqL3uWL8qyFiqsm8fCfPB1bxVMznfokNirjXsI9v8
S+NPjv5y1y/BwV1MD3chD4fHitC1vtCHS13pPyKGQMWegDFYlINp5n4EJgxa2X9k
V5QHsIYmWLFsuPfuwv63ICISqbb0M7X6Vav1BZekNBqXS1qjkeG2VOHVXrUAEQEA
AYkCNgQYAQgAIBYhBEqLH7Dir00EH72CnGONSp1M9rvHBQJnx12MAhsMAAoJEGON
Sp1M9rvHCKgP/ingWTqDWIred4ogKcTqbt0C0vMtmnqpgjhMmoIS9PCdpcAWgRwo
Mdr7BD+SUTnWEm97txarlusCUgG+1oQk92hAt9mI4PGbhe1m1gm5PnaMP/yd84v0
05r0jtB3RmZi4D/ut86fnG2ANSoYp8Pg9ic93Fda7wK9Zhj1NePxDjqtXFNDZW/
AXeXJWYTKhGUaoKrGwJ4h5DJzIPF/tMGBCbPokeR78TdZGk0EDFLV3aGt7Z5+01k
r1FSUi7kxRlqaD8HxbhVbgH69rGXJuuI0/vETEA/SVRoudAX+/3HeyA2juq/biA
qZhvLSXPDXCMBcsfzxdqNAIQ/1RdLqG1YJFX8rwhUUnsCqwIcHS5EH8vVgFuc2f
Vs1f21RS2491pcTU7f+cb3tqew8cgHKJl0Xh1fukHb1Uevr3IX6sk9shwtC30GTN
Ue726uW7STs4Wl0FE2RXFnINFmFkvSIOL/VdYDgChvx1YneGz7/3RcSsq1P0Byuys
/khAWkawWxjmb1fEZ4JCwcZY513WFOyZxuaHU+0aT/c8LgtJQIS3+j2uyDHx63XC
XqLtgY+qaPa6Y0s+DYV9WP1I16SsQ+q1y3pSKfIH7Ibj60wCnHAJ+hvInzG+g+P
yRnZAat2qdczN2Zul855GCN3KQtW14n+I/o0S1nZ1fAE6i2kY6y/THrr
=0nyq
-----END PGP PUBLIC KEY BLOCK-----
grkurushin@grkurushin:~$
[3] 0: bash* "grkurushin" 23:18 04-мар-25
```

Рис. 4.3: Вывод ключа в терминал

4.3.1 Добавление ключа на ГитХаб.

Скопировав ключ, переносим его на ГитХаб, создаем на сайте новый ключ и вставляем скопированный ключ в необходимое поле. (рис. 4.4)

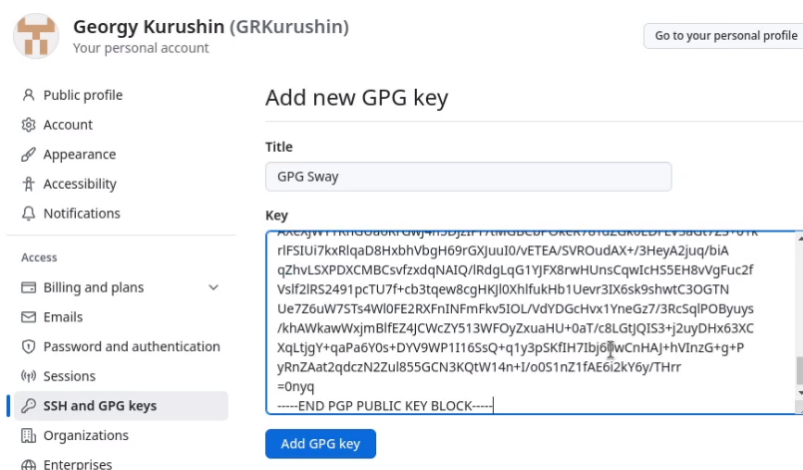


Рис. 4.4: Новый ключ PGP

4.3.2 Настройка автоматических подписей коммитов git

При помощи команд `git config --global user.signingkey`, `git config --global commit.gpgsign true` и `git config --global gpg.program $(which gpg2)` самостоятельно выбираем подписи коммитов в git. (рис. 4.5)

```
foot
--unset          remove a variable: name [<value-pattern>]
--unset-all      remove all matches: name [<value-pattern>]
--rename-section переименовать раздел: старое-имя новое-имя
--remove-section удалить раздел: имя
-l, --list       показать весь список
-e, --edit       открыть в редакторе
--get-color      find the color configured: slot [<default>]
--get-colorbool  find the color setting: slot [<stdout-is-tty>]

Display options
-z, --[no-]null    завершать значения НУЛЕВЫМ байтом
--[no-]name-only   показывать только имена переменных
--[no-]show-origin показать источник настройки (файл, стандартный ввод, двоичный объект, командная строка)
--[no-]show-scope  show scope of config (worktree, local, global, system, command)
--[no-]show-names  show config keys in addition to their values

Тип
-t, --[no-]type <тип> value is given this type
--bool          значение – это «true» (правда) или «false» (ложь)
--int           значение – это десятичное число
--bool-or-int   значение – это --bool или --int
--bool-or-str   value is --bool or string
--path          значение – это путь (к файлу или каталогу)
--expiry-date   значение – это дата окончания срока действия

Другое
--[no-]default <value> with --get, use default value when missing entry
--[no-]comment <value> human-readable comment string (# will be prepended as needed)
--[no-]fixed-value use string equality when comparing values to value pattern
--[no-]includes    учитывать директивы include (включения файлов) при запросе

grkurushin@grkurushin:~$ git config --global user.singingkey 4AA51FB0E2AF43841FBD829C638D4A994CF6BBC7
grkurushin@grkurushin:~$ git config --global commit.gpgsign true
grkurushin@grkurushin:~$ git config --global gpg.program $(which gpg2)
grkurushin@grkurushin:~$ 
[3] 0: bash* "grkurushin" 23:23 04-мар-25
```

Рис. 4.5: Настройка необходимых подписей коммитов

4.4 Настройка gh

Введя в терминал команду `gh auth login`, ответим на необходимые в терминале вопросы, после чего авторизуемся через браузер. (рис. 4.6)

```
foot
Enter file in which to save the key (/home/grkurushin/.ssh/id_rsa):
Created directory '/home/grkurushin/.ssh'.
Enter passphrase for "/home/grkurushin/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/grkurushin/.ssh/id_rsa
Your public key has been saved in /home/grkurushin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:IPYHESIwzUSEDgCofq5TFeiSEoLQQ0sSWZ5wUgvZ+04 grkurushin@grkurushin
The key's randomart image is:
+---[RSA 4096]-----+
|#0+ +0= . o. |
|*0+=.oo. o . |
|=0*+ . + . |
|+.... o . . |
|+.... S . . |
|.. oE . . |
| +o . . |
| . . . |
|.o . |
+----[SHA256]-----+
grkurushin@grkurushin:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/grkurushin/.ssh/id_rsa):
/home/grkurushin/.ssh/id_rsa already exists.
Overwrite (y/n)?
grkurushin@grkurushin:~$ gh auth login
? Where do you use GitHub? 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: 045F-C436
Press Enter to open https://github.com/login/device in your browser...
Окно или вкладка откроются в текущем сеансе браузера.
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as GRKURUSHIN
grkurushin@grkurushin:~$ █
[3] 0:bash* "grkurushin" 23:28 04-мар-25
```

Рис. 4.6: Настройка gh и авторизация в браузере

4.5 Создание и настройка репозитория курса.

Используя команды `mkdir`, `gh repo`, `create study` и `git clone` создаем репозиторий курса. (рис. 4.7)

```
foot
Получение объектов: 100% (36/36), 19.37 КиБ | 3.23 МиБ/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»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/grkurushin/work/study/2024-2025/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 594.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/grkurushin/work/study/2024-2025/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 1.13 МиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы$ cd os-intro/
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ rm package.json
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COU
RSE
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ make
Usage:
  make <target>

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

grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ make prepare
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$
[3] 0: bash* "grkurushin" 23:34 04-мар-25
```

Рис. 4.7: Созданный репозиторий, папка первой лабораторной работы

Отправляем файлы первой лабораторной работы на сервер. (рис. 4.8)


```

foot
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.p
y
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/.projectile
create mode 100644 project-personal/stage6/presentation/.texlabroot
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/placeimg_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.p
y
create mode 100644 project-personal/stage6/report/report.md
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 100% (40/40), готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.31 Киб | 20.14 Миб/с, готово.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/GRKURUSHIN/study_2024-2025_os-intro.git
   e90364f..4420926  master -> master
grkurushin@grkurushin:~/work/study/2024-2025/Операционные системы/os-intro$ 
[3] 0: bash* "grkurushin" 23:43 04-мар-25

```

Рис. 4.8: Отправка файлов на сервер

5 Контрольные вопросы

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

Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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. Системы контроля версий (Version Control System, VCS) применяются для:

- Хранение полной истории изменений причин всех производимых изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок в программе
- Совместная работа группы над одним проектом
- Возможность изменять код, не мешая работе других пользователей

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

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

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

Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): Одно основное хранилище всего проекта Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно Децентрализованные VCS (Git; Mercurial; Bazaar): У каждого пользователя свой вариант (возможно не один) репозитория Присутствует возможность добавлять и забирать изменения из любого репозитория . В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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

Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Участник проекта (пользователь) перед началом работы посредством определенных команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.

6. Каковы основные задачи, решаемые инструментальным средством git?

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.

Наиболее часто используемые команды git: • создание основного дерева репозитория: `git init` • получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` • отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` • просмотр списка изменённых файлов в текущей директории: `git status` • просмотр текущих изменений: `git diff` • сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add`. – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` • удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` • сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` – сохранить добавленные изменения с внесением комментария через встроенный редактор `git commit` • создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` • 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) • отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` • слияние ветки с текущим деревом: `git merge --no-ff имя_ветки` • удаление ветки: – удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` – принудительное удаление локальной ветки: `git branch -D имя_ветки` – удаление ветки с центрального репозитория: `git push origin :имя_ветки`

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

`git push --all (push origin master/любой branch)`

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

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть

главная ветка (master), или ствол (trunk). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

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 в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

6 Выводы

В результате выполнения данной лабораторной работы я приобрел необходимые навыки работы с гит, научился созданию репозитория, gpg и ssh ключей, настроил каталог курса и авторизовался в gh.

Список литературы