

Лабораторная работа №2

Архитектура операционных систем

Касымова Элина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Контрольные вопросы	18
6	Выводы	26
	Список литературы	27

Список иллюстраций

4.1	Операционные системы	10
4.2	Клонирование	11
4.3	Репозиторий	11
4.4	Имя	12
4.5	Email	12
4.6	Ключ	12
4.7	Создание	13
4.8	Список ключей	13
4.9	Экспорт	14
4.10	Копирование	14
4.11	Копирование	15
4.12	Код	15
4.13	GPG	16
4.14	КАталогии	16
4.15	Commit	16
4.16	Git	16
4.17	Git add, commit	17
4.18	Git push	17

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	8
-----	---	---

1 Цель работы

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

Цель данного шаблона — максимально упростить подготовку отчётов по лабораторным работам. Модифицируя данный шаблон, студенты смогут без труда подготовить отчёт по лабораторным работам, а также познакомиться с основными возможностями разметки Markdown.

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 Теоретическое введение

Системы контроля версий. Общие понятия

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.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

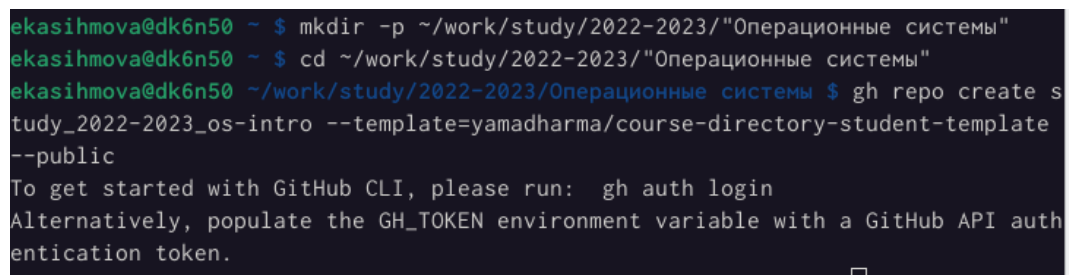
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям

Имя каталога	Описание каталога
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

1)Создаем репозиторий для предмета “Операционные системы”.

A terminal window with a dark background and light-colored text. The user 'ekasihmova@dk6n50' is shown at the prompt. The commands executed are: 'mkdir -p ~/work/study/2022-2023/"Операционные системы"', 'cd ~/work/study/2022-2023/"Операционные системы"', and 'gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public'. The output shows the repository creation details and instructions for using GitHub CLI.

```
ekasihmova@dk6n50 ~ $ mkdir -p ~/work/study/2022-2023/"Операционные системы"
ekasihmova@dk6n50 ~ $ cd ~/work/study/2022-2023/"Операционные системы"
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ gh repo create s
tudy_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
To get started with GitHub CLI, please run:  gh auth login
Alternatively, populate the GH_TOKEN environment variable with a GitHub API auth
entication token.
```

Рис. 4.1: Операционные системы

```

entification token.
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github.com:EllinaKasymowa/study_2022-2023_arh-pc.git
Клонирование в «study_2022-2023_arh-pc»...
remote: Enumerating objects: 196, done.
remote: Counting objects: 100% (196/196), done.
remote: Compressing objects: 100% (178/178), done.
remote: Total 196 (delta 28), reused 170 (delta 13), pack-reused 0
Получение объектов: 100% (196/196), 10.26 МиБ | 1.61 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Updating files: 100% (214/214), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/e/k/ekasihmova/work/study/2022-2023/Операционные системы/study_2022-2023_arh-pc/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 1.13 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/e/k/ekasihmova/work/study/2022-2023

```

Рис. 4.2: Клонирование

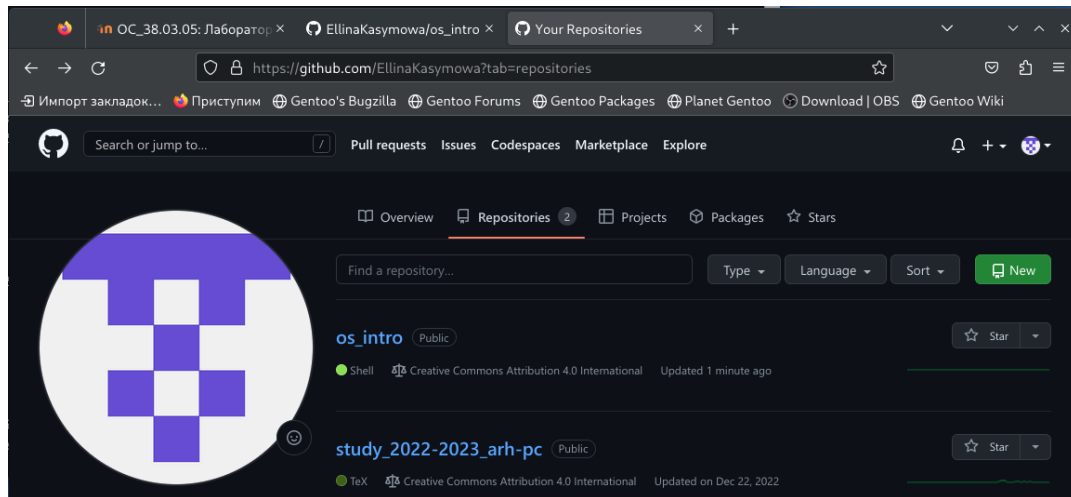


Рис. 4.3: Репозиторий

2) Задаю имя и email своего репозитория

```
kasimova@dk6n50 ~/work/study/2022-2023/Операционные системы $ git config --global user.name "EllinaKasymowa"
```

Рис. 4.4: Имя

```
kasimova@dk6n50 ~/work/study/2022-2023/Операционные системы $ git config --global user.email "kasymowa2004@icloud.com"
```

Рис. 4.5: Email

3) Генерирую ключ.

```
kasimova@dk6n50 ~/work/study/2022-2023/Операционные системы $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 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:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
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.
```

Рис. 4.6: Ключ

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: EllinaKasymowa
Адрес электронной почты: kasymowa2004@icloud.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "EllinaKasymowa <kasymowa2004@icloud.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
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: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/e/k/ekasihmova/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/e/k/ekasihmova/.gnupg/openpgp-revocs.d/AF991BC67BD0F6930D75886260CD60BEF6320FC4.rev'.
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2023-02-16 [SC]
     AF991BC67BD0F6930D75886260CD60BEF6320FC4
uid          EllinaKasymowa <kasymowa2004@icloud.com>

```

Рис. 4.7: Создание

4) Вывожу список ключей и копирую отпечаток приватного ключа

```

ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ 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
/afs/.dk.sci.pfu.edu.ru/home/e/k/ekasihmova/.gnupg/pubring.kbx
-----
sec  rsa4096/60CD60BEF6320FC4 2023-02-16 [SC]
     AF991BC67BD0F6930D75886260CD60BEF6320FC4
uid          [ абсолютно ] EllinaKasymowa <kasymowa2004@icloud.com>
ssb  rsa4096/21ACD81C4413DE01 2023-02-16 [E]

```

Рис. 4.8: Список ключей

5) Экспортирую ключ в формате ASCII по его отпечатку.

```

ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ gpg --armor --export
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGPuUjYBEADPurZbguYmdxqM83gLeBChjbl8B5Izq/gwxCdLr8sPXMzv2lW0
G3Z8eMAXZPU6JP80QzR5b5R0I/h+nXFPZOCUr/6heC/nQRp5lb3GFtUqb87q15W6
sPr1LtGlrucFInf+wyw3+pEIpEaDw8gv1MR/z6SWg5oLHwbEdrNjMmPc849g8Rj
12Mmw1TG2PH+b0oJb+ksW7AF1rjqcmaTS5d2tmCVioM1ZtnSlybt0HtiLVP0fLS
v0DFUBRJveXDb1XjwRCob4ACmBgYTz3sabPNfjV7MERHvJj4tXB5YtyVu+if+RJr
iGJwmPYuFe3umupAItKoGD5PcrHtY5+Hgff1f5HofNyZxmb/HrTxGFTT9DPKwImi
uu5FTn2129rrVYpIQpC9Ln1fkarWSPIS0xqHexD8kzVVGsa1uR4pEaC5Wf+1N+a1
cY0vGQ1jkCgmRUlpe5hRiTy305D/rCbAVIyHmKeyQ9xuBKj0YApLswTksZy5JlN
IrcC3bvLCWJT10TfgVj2XZLCrIoGYZGC5qX0StlvFJfa1FQR4I63QAzcXfGIKjsR
H7FJQu+CLCmHqZZEUvBaCANF2eQj3DPhl1Etw9FZQcQDtd2zo39o8/me3Km1amJU
CoiPbW6SLi8azB+jgzBuCrBj3c2U1qbZcz54xxonX3K83ggHXXyI/NLBvQARAQAB
tChFbGxpbmFLYXN5bW93YSA8a2FzeWlvd2EyMDA0QG1jbG91ZC5jb20+iQJOBBMB
CAA4F1EEr5kbnvQ9pMNdYh1YM1gvvYyD8QFamPuUjYCGwMFCwkIBwIGFQoJCA5C
BBYCAwECHGECF4AACgkQYMIgvvYyD8SiMBAAPrg93Dw1aXJbRL/Y9ied9v1Hiq
m3Nxt1C2/kx1NAJAu/5X3bG5DwXrQn4aAaP9zQXcuNptRuOH1kR6C00ho8PmPY0
mJaHD0eyK124GZ2S7JKsb3D2heWqGHd2EeSVOWYsW+0NxK2m/1MBtwXU0w+4qoD3
4G99f1x0EjuVwo73tjQIFX/JyNLkCIWSQS/UPYpZ9H2zqH407kS0ysq1mqSNB/T
Rt8rVTh3VHVJxJ6Yr1SsnYmpjVCq25sVMR2TiiMbm8zQ8+Gg/FVvb71KQ7soNVxo
RsuHk0K6sCbl6QAEEnlUmwnW6v0j61pMBIgnq0594MbYwBNC+6iJRSExgoaTOHRc

```

Рис. 4.9: Экспорт

6) Копирую свой сгенерированный PGP ключ в буфер обмена.

```

ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ gpg --armor --export | xclip -sel clip
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $

```

Рис. 4.10: Копирование

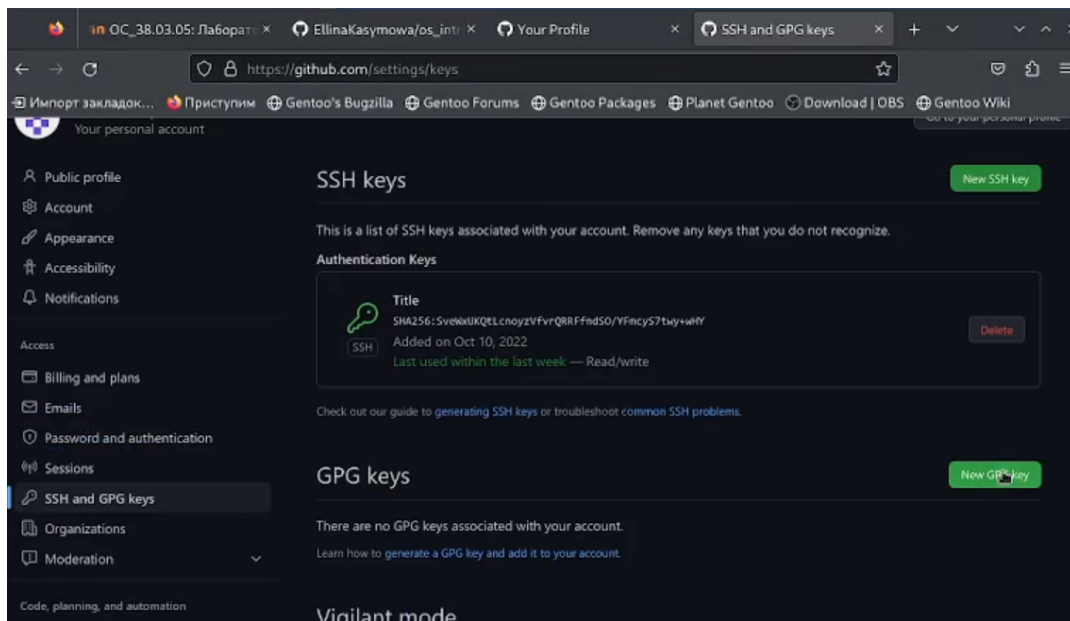


Рис. 4.11: Копирование

7) Вставляю код в New GPG key.

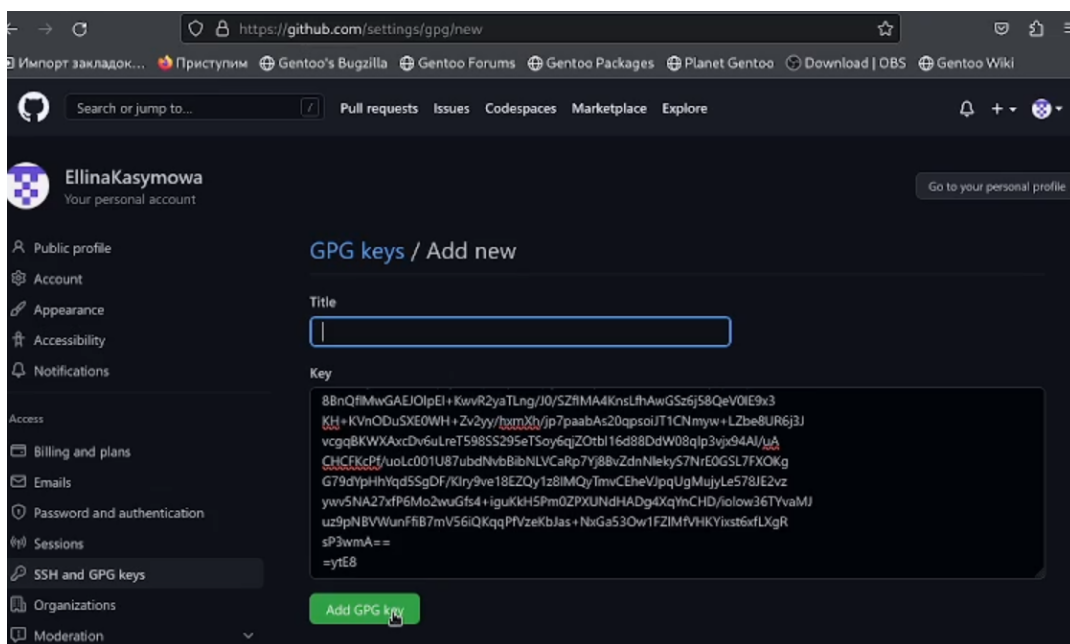


Рис. 4.12: Код

8) Мы получили GPG ключ.

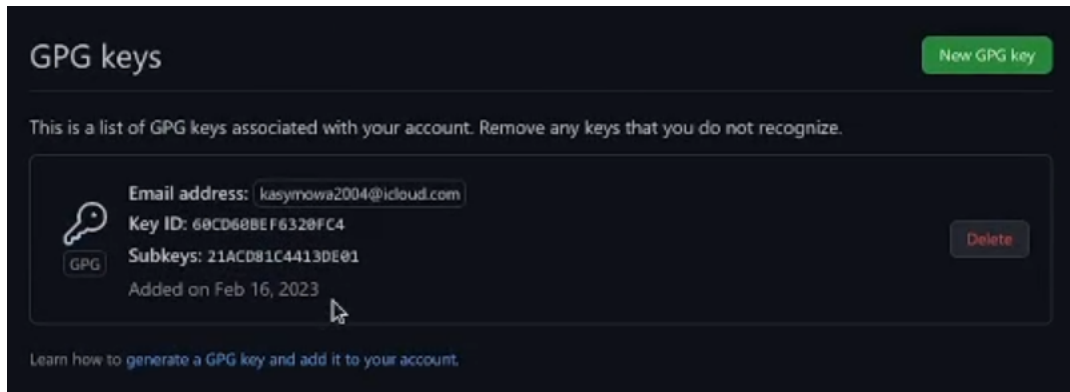


Рис. 4.13: GPG

9) Далее переходим в os-intro и создаю необходимые каталоги, коммитим

```
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы $ cd ~/work/study/2022-2023/Операционные системы/os-intro
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro > COURSE
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ make
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $
```

Рис. 4.14: КАталоги

```
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os_intro $ git commit -a -m 'your commit
message'
[master 8383487] your commit message
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
```

Рис. 4.15: Commit

10) Используя введённый email, указываю Git, применяемый при подписи коммитов.

```
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os_intro $ git config --global user.signin
gkey
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os_intro $ git config --global commit.gpgs
ign true
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os_intro $ git config --global gpg.program
$(which gpg2)
```

Рис. 4.16: Git

11) Отправляю файлы на сервер.


```

ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ git add .
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'lab02'
bash: it: команда не найдена
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'lab02'
[master c1dd93b] lab02
360 files changed, 100327 insertions(+), 1 deletion(-)
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-numeric.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
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab02/report/report.md

```

Рис. 4.17: Git add, commit

```

ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $ git push
Перечисление объектов: 41, готово.
Подсчет объектов: 100% (41/41), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (39/39), 343.06 КиБ | 2.52 МиБ/с, готово.
Всего 39 (изменений 4), повторно использовано 1 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:EllinaKasymowa/os_intro.git
 8383487..c1dd93b master -> master
ekasihmova@dk6n50 ~/work/study/2022-2023/Операционные системы/os-intro $

```

Рис. 4.18: Git push

5 Контрольные вопросы

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

Система контроля версий (VCS) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа. Если вы графический или веб- дизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.

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

Хранилище-система, которая обеспечивает хранение всех существовавших вариантов файлов Commit-фиксация изменений История-список предыдущих ревизий Рабочая копия-копия другой ветки Команде commit можно передать сообщение, описывающее изменения в ревизии. Она также записывает идентифи-

катор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -m, или – message. Можно также вводить сообщения, состоящие из нескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. commit -m “добавлен первый файла.

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

Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная система контроля версий Mercurial. Существуют СКВ централизованные, в которых имеется один репозиторий, в который собираются изменения со всех рабочих копий разработчиков, и децентрализованные, когда репозитория много, и они могут обмениваться изменениями между собой. Централизованные СКВ – репозиторий один. У каждого разработчика своя рабочая копия. Время от времени разработчик может затягивать к себе в рабочую копию новые изменения из репозитория, или проталкивать свои изменения из своей рабочей копии в репозиторий. Прочие особенности централизованных СКВ зависят от реализации.

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

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может ис-

пользовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

5)Опишите порядок работы с общим хранилищем VCS.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельт компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые

системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

6) Каковы основные задачи, решаемые инструментальным средством git?

Устанавливает единственную новую команду, git. Все возможности предоставляются через подкоманды этой команды. Вы можете просмотреть краткую справку командой help. Некоторые идеи группируются по темам, используйте help topics для списка доступных тем. Одна из функций системы контроля версий — отслеживать кто сделал изменения. В распределенных системах для этого требуется идентифицировать каждого автора уникально в глобальном плане. Большинство людей уже имеют такой идентификатор: email адрес. Vazaar достаточно умен, чтобы автоматически создавать email адрес из текущего имени и адреса хоста. Основные задачи: создание ветки, размещение веток, просмотр изменений, фиксация изменений, сообщение из текстового редактора, выборочная фиксация, удаление зафиксированных изменений, игнорирование файлов, просмотр истории, статистика ветки, контроль файлов и каталогов, ветвление, объединение веток, публикация ветки.

7) Назовите и дайте краткую характеристику командам git.

Обновление рабочей копии По мере внесения изменений в проект рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта увеличивается. Это повышает риск возникновения конфликтных изменений (см. ниже). Поэтому удобно поддерживать рабочую копию в состоянии, максимально близком к текущей основной версией, для чего разработчик выполняет операцию обновления рабочей копии (update) насколько возможно часто (реальная частота обновлений определяется частотой внесения изменений, зависящей от активности разработки и числа разработчиков, а также временем, затрачиваемым на каждое обновление — если оно велико, разработчик вынуж-

ден ограничить частоту обновлений, чтобы не терять время). Модификация проекта Разработчик модифицирует проект, изменяя входящие в него файлы в рабочей копии в соответствии с проектным заданием. Эта работа производится локально и не требует обращений к серверу VCS. Фиксация изменений Завершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер (либо в основную ветвь, если работа над заданием полностью завершена, либо в отдельную ветвь разработки данного задания). VCS может требовать от разработчика перед фиксацией обязательно выполнить обновление рабочей копии. При наличии в системе поддержки отложенных изменений (shelving) изменения могут быть переданы на сервер без фиксации. Если утверждённая политика работы в VCS это позволяет, то фиксация изменений может проводиться не ежедневно, а только по завершении работы над заданием; в этом случае до завершения работы все связанные с заданием изменения сохраняются только в локальной рабочей копии разработчика.

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

Мы создаем новую ветку выполнив `git init` в уже созданном каталоге: `% mkdir tutorial % cd tutorial % ls -a ./ ./ % pwd /home/mbp/work/bzr.test/tutorial % % git init % ls -aF ./ ./ .git/ %` Мы обычно обращаемся к веткам на нашем компьютере просто передав имя каталога содержащего ветку. bzr также поддерживает доступ к веткам через `http` и `sftp`, например: `git log http://bazaar-vcs.org git // git.dev/ git log sftp://bazaarvcs.org/bzr/bzr.dev/` Установив для `git` плагины можно также осуществлять доступ к веткам с использованием `rsync`. Команда `status` показывает какие изменения были сделаны в рабочем каталоге с момента последней ревизии: `% git status modified: foo bzr status` скрывает неинтересные файлы, которые либо не менялись, либо игнорируются. Также команде `status` могут быть переданы необязательные имена файлов, или каталогов для проверки. Команда `diff` показывает изменения в тексте файлов в стандартном формате `diff`. Вывод этой команды может быть передан другим командам, таким как `"patch"`, `"diffstat"`, `"filterdiff"`

и "colordiff": % git diff === added file 'hello.txt' — hello.txt 1970-01-01 00:00:00 +0000 +++ hello.txt 2005-10-18 14:23:29 +0000

6.2. Указания к лабораторной работе

75 @@ -0,0 +1,1 @@ +hello world

Команде commit можно передать сообщение описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. git commit -m "добавлен первый файл" Если вы передадите список имен файлов, или каталогов после команды commit, то будут зафиксированы только изменения для переданных объектов. Например: bzip commit -m "исправления документации" commit.py Если вы сделали какие-либо изменения и не хотите оставлять их, используйте команду revert, что бы вернуться к состоянию предыдущей ревизии. Многие деревья с исходным кодом содержат файлы которые не нужно хранить под контролем версий, например резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать git игнорировать их добавив их в файл .ignore в корне рабочего дерева. Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду ignored: % ignored config.h ./config.h configure.in ~ *~ log Команда bzip log показывает список предыдущих ревизий. Команда log -forward делает тоже самое, но в хронологическом порядке, показывая более поздние ревизии в конце может контролировать файлы и каталоги, отслеживая переименования и упрощая их последующее объединение: % mkdir src % echo 'int main() {}' > src/simple.c % add src added src added src/simple.c % status added: src/ src/simple.c bzip remove удаляет файл из под контроля версий, но может и не удалять рабочую копию файла². Это удобно, когда вы добавили не тот файл, или решили, что файл на самом деле не должен быть под контролем версий. % rm -r src % remove -v hello.txt ? hello.txt % status removed: hello.txt src/ src/simple.c unknown: hello.txt Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия

может быть потенциальной новой веткой. Если две ветки разошлись (обе имеют уникальные изменения) тогда `merge` — это подходящая команда для использования. Объединение автоматически вычислит изменения, которые существуют на объединяемой ветке и отсутствуют в локальной ветке и попытается объединить их с локальной веткой. `git merge URL`.

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

Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой эта команда называется `branch`: Управление версиями `git branch`
`cd git.dev` Эта команда копирует полную историю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки.

10) Как и зачем можно игнорировать некоторые файлы при `commit`?

Нет проблем если шаблон для игнорирования подходит для файла под контролем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют показывающиеся неизвестные файлы, или просто игнорируются. Файл `git.rignore` обычно должен быть под контролем версий, что бы новые копии ветки видели такие же шаблоны: `git add . gitignore git commit -m "Добавлены шаблоны для игнорирования"`. Многие деревья с исходным кодом содержат файлы, которые не нужно хранить под контролем версий, например, резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать `bzr` игнорировать их добавив их в файл в корне рабочего дерева. Этот файл содержит список шаблонов файлов, по одному в каждой строчке. Обычное содержимое может быть таким: `.o ~ .tmp .py [со]` Если шаблон содержит слеш, то он будет сопоставлен с полным путем начиная от корня рабочего дерева; иначе он сопоставляется только с именем файла. Таким образом пример

выше игнорирует файлы с расширением .o во всех подкаталогах, но пример ниже игнорирует только config.h в корне рабочего дерева и HTML файлы в каталоге doc/: ./config.h doc/.html Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду *git ignored* : *\$ git ignored config.h ./config.h configure.in~ ~ \$*

6 Выводы

Прodelав данную лабораторную работу мы изучили идеологию и применение средств контроля версий и освоили умение по работе с git.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.