

Лабораторная работа 2

Система контроля версий Git

Елисейкина Надежда Михайловна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	12
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Сайт github.com	12
4.2	Ввод имени владельца	13
4.3	Ввод команды почты владельца	14
4.4	Настройка utf-8 в выводе сообщений git	15
4.5	Имя начальной ветки	15
4.6	Параметр autocrlf	16
4.7	Параметр safecrlf	16
4.8	Копирование ключа	17
4.9	Ключ	17
4.10	Создание каталога «Архитектура компьютера»	18
4.11	Создание репозитория	18
4.12	Переход в каталог	19
4.13	Клонирование репозитория	20
4.14	Ввод команды для удаления	20
4.15	Ввод команды для создания каталогов	21
4.16	Отправка данных на сервер	22
4.17	Проверка правильности создания рабочего пространства	22
4.18	Создание каталога	23
4.19	Загрузка файлов на github	24
4.20	Проверка файлов на github	24

Список таблиц

3.1 Основные команды git 9

1 Цель работы

Изучить идеологию и применение средств контроля версии. Приобрести практические навыки по работе с git.

2 Задание

1. Открыть сайт <https://github.com/>, создать учётную запись, заполнить основные данные.
2. Создать предварительную конфигурацию git. Открыть терминал и ввести необходимые команды, указав имя и email владельца репозитория.
3. Настроить utf-8 в выводе сообщений git.
4. Задать имя начальной ветки (master).
5. Настроить параметр autocrlf.
6. Настроить параметр safecrlf.
7. Сгенерировать пару ключей (приватный и открытый).
8. Скопировать из локальной консоли ключ в буфер обмена.
9. Внести ключ на сайте, указав имя ключа Title.
10. Создать каталог для предмета «Архитектура компьютера».
11. Создать репозиторий через web-интерфейс github.
12. Клонировали созданный репозиторий.
13. Перейти в каталог курса.
14. Удалить лишнее файлы.
15. Создать необходимые каталоги.
16. Отправить необходимые файлы на сервер.
17. Проверить правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

Задание для самостоятельной работы

1. Создать каталог для внесения отчета по выполнению лабораторной работы

в (labs>lab02>report).

2. Создать каталог для внесения предыдущих лабораторных работ.
3. Скопировать отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
4. Загрузить и проверить файлы на github.

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 за-

блокировать 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.

Система контроля версий Git Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

Основные команды git

В табл. [3.1] приведены наиболее часто используемые команды git.

Таблица 3.1: Основные команды git

Имя каталога	Описание каталога
<code>git</code>	создание основного дерева репозитория
<code>init</code>	

Имя каталога	Описание каталога
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 изменения
git add	имена_файлов добавить конкретные изменённые и/или созданные файлы и/или каталоги
git rm	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
git add .	добавить все изменённые и/или созданные файлы и/или каталоги

Стандартные процедуры работы при наличии центрального репозитория

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

```
git checkout master git pull git checkout -b имя_ветки
```

Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту: `git status` и при необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий.

Затем полезно просмотреть текст изменений на предмет соответствия прави-

лам ведения чистых коммитов: `git diff`

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями: `git add имена_файлов` `git rm имена_файлов`

Если нужно сохранить все изменения в текущем каталоге, то используем: `git add .`

Затем сохраняем изменения, поясняя, что было сделано: `git commit -am "Some commit message"` и отправляем в центральный репозиторий: `git push origin имя_ветки` или `git push`

4 Выполнение лабораторной работы

1. Зашли на сайт <https://github.com/>, создали учётную запись и заполнили данные (рис. 4.1 Сайт github.com).

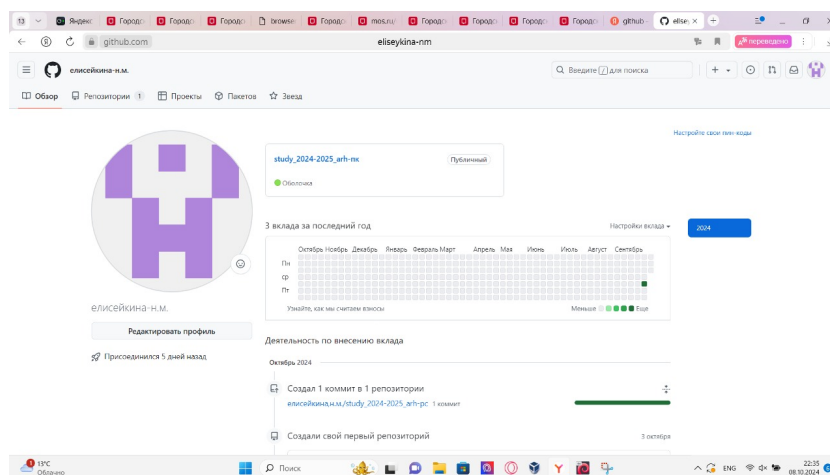


Рис. 4.1: Сайт github.com

2. Создали предварительную конфигурацию git. Ввели следующие команды, указав имя владельца репозитории (рис. 4.2 Ввод имени владельца).

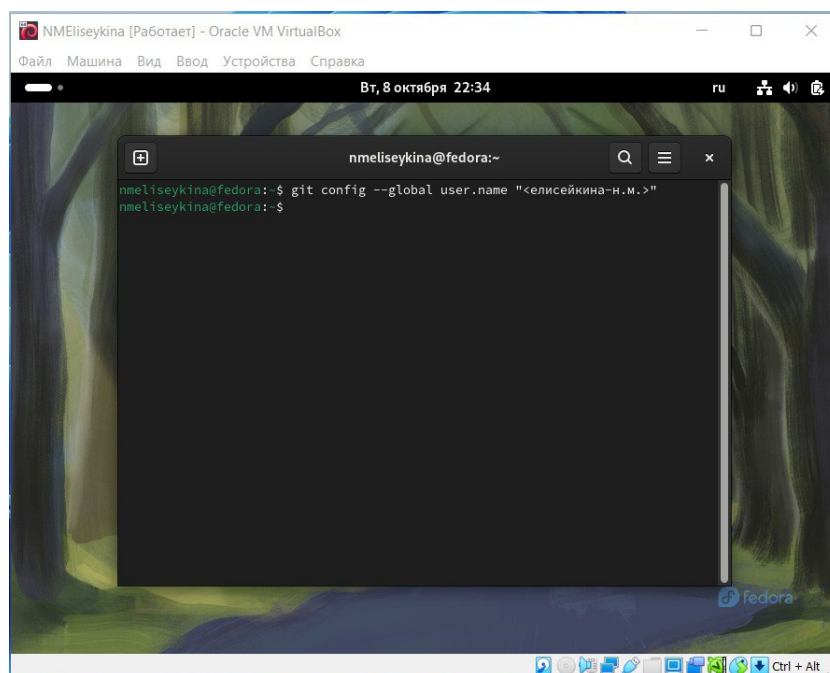


Рис. 4.2: Ввод имени владельца

Ввели следующие команды, указав email владельца репозитории (рис. 4.3 Ввод команды почты владельца).

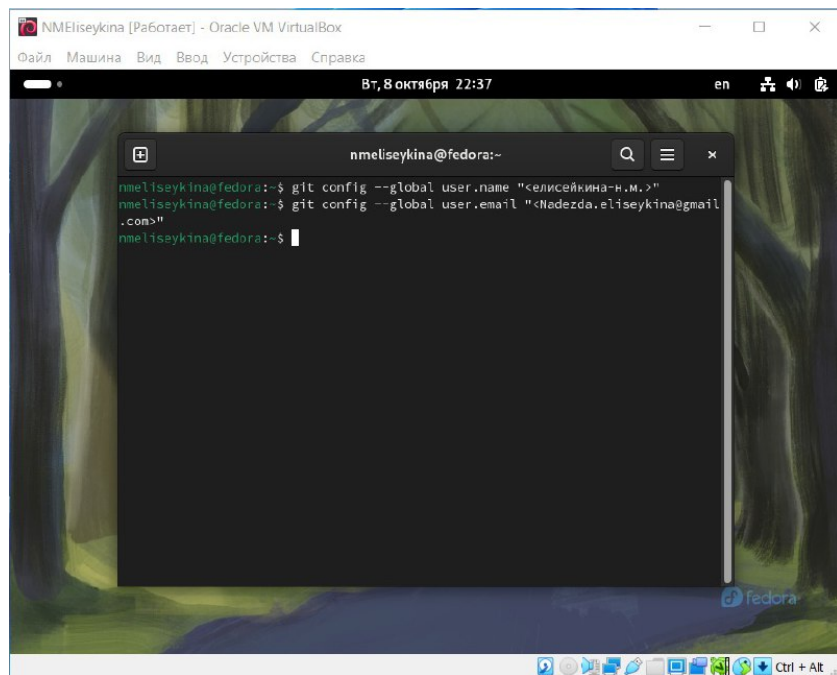


Рис. 4.3: Ввод команды почты владельца

3. Настроим utf-8 в выводе сообщений git (рис. 4.4 Настройка utf-8 в выводе сообщений git).

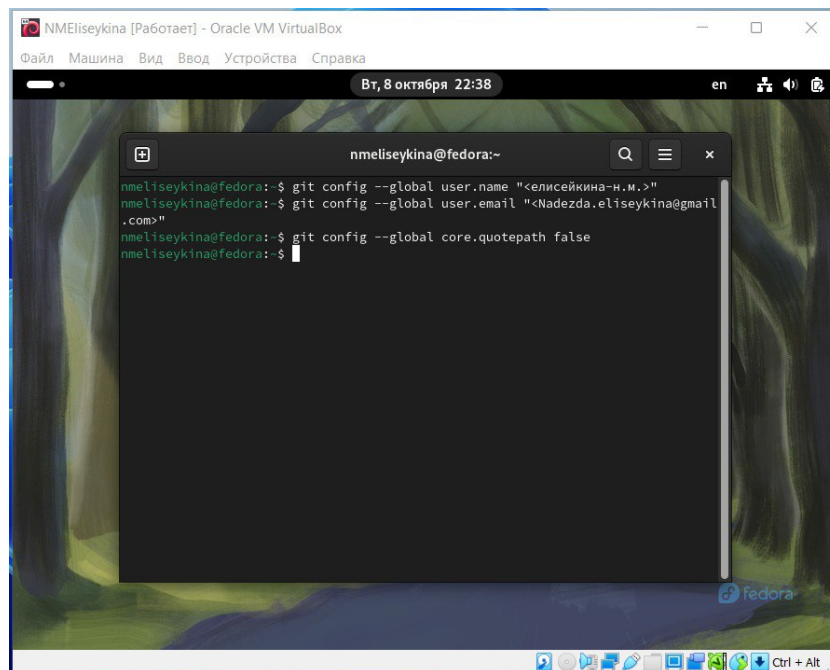


Рис. 4.4: Настройка utf-8 в выводе сообщений git

4. Задали имя начальной ветки (рис. 4.5 Имя начальной ветки).

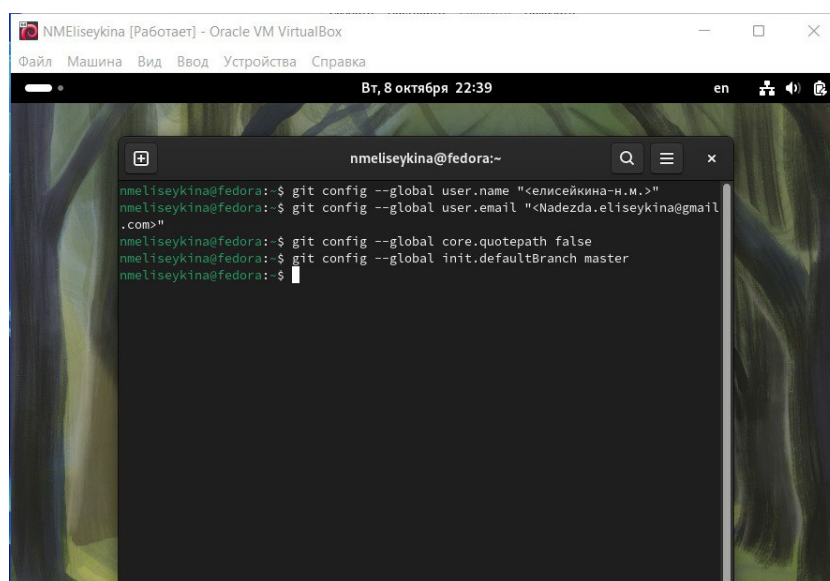


Рис. 4.5: Имя начальной ветки

5. Настроили autocrlf (рис. 4.6 Параметр autocrlf).

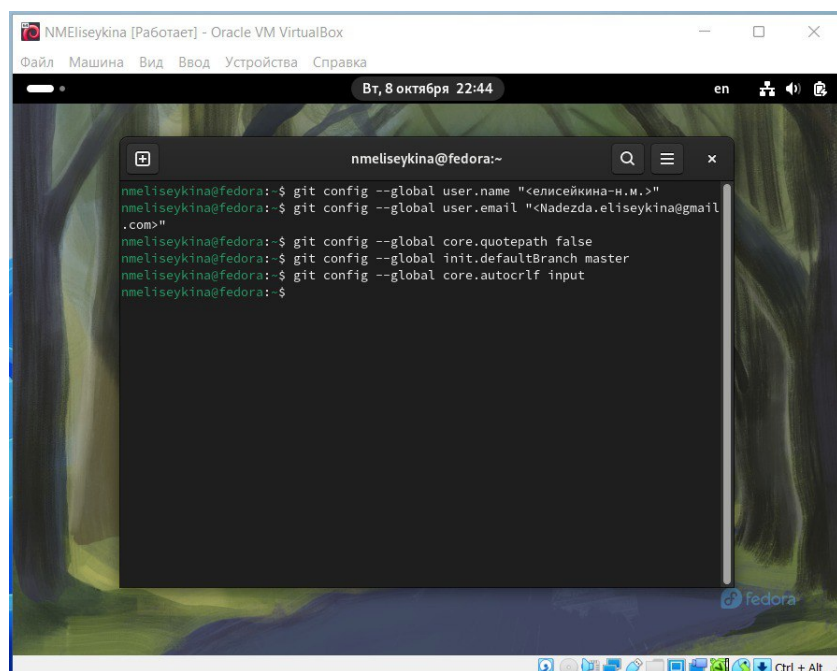


Рис. 4.6: Параметр autocrlf

6. Настроили safecrlf (рис. 4.7 Параметр safecrlf).

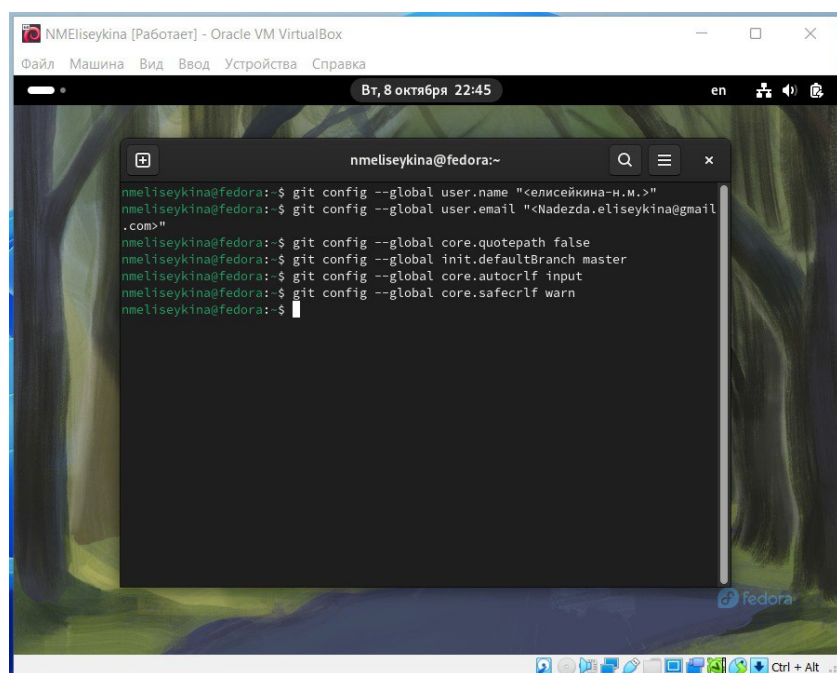


Рис. 4.7: Параметр safecrlf

7. Создание SSH ключа Сгенерировали пару ключей

Скопировали из локальной консоли ключ (рис. 4.8 Копирование ключа).

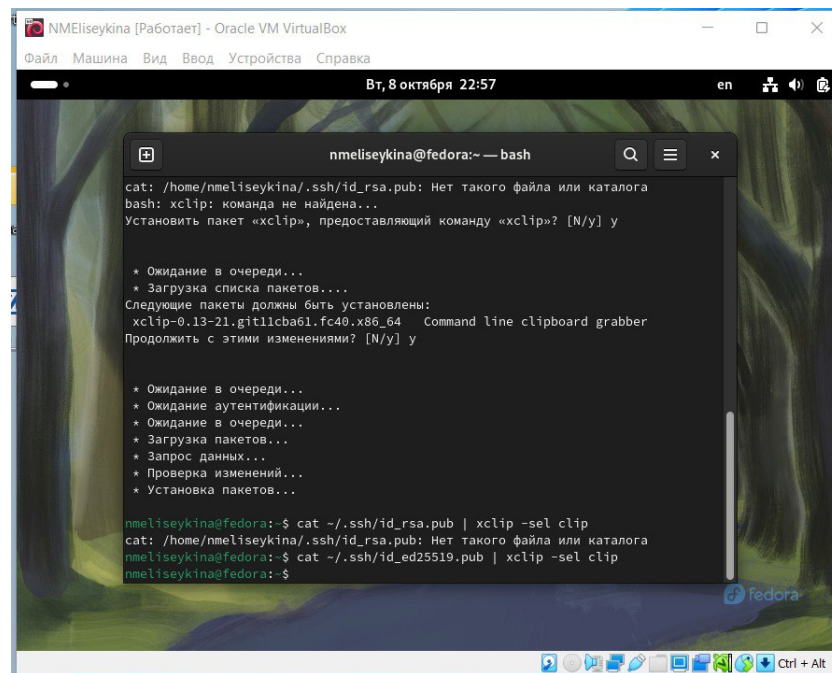


Рис. 4.8: Копирование ключа

8. Внесли ключ на сайте (рис. 4.9 Ключ).

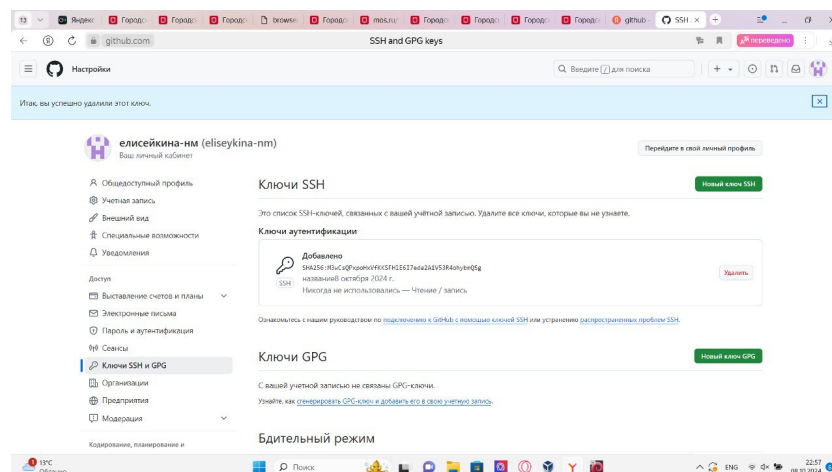


Рис. 4.9: Ключ

9. Создали каталог для предмета «Архитектура компьютера» (рис. 4.10 Создание каталога «Архитектура компьютера»).

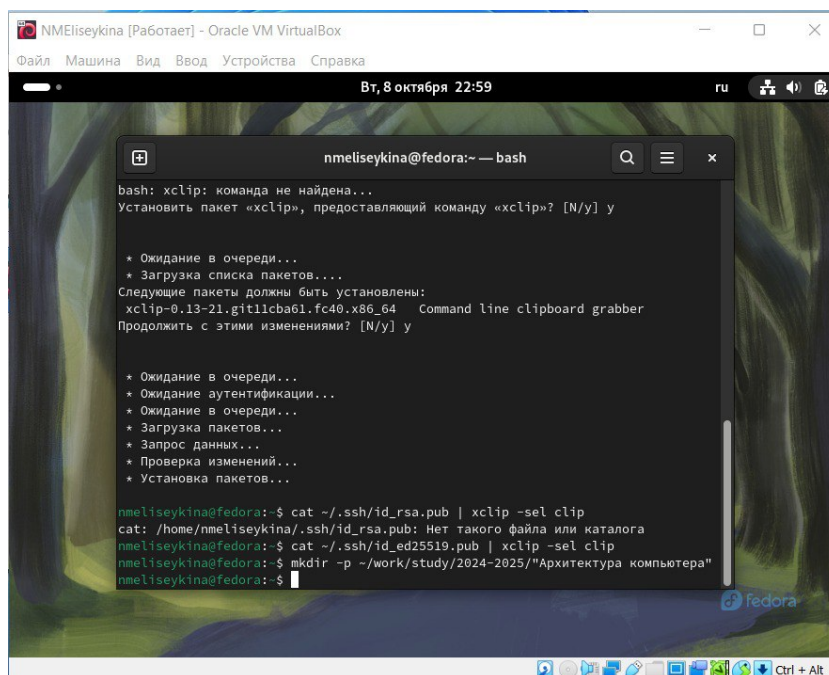


Рис. 4.10: Создание каталога «Архитектура компьютера»

10. Создали репозиторий через сайт github (рис. 4.11 Создание репозитория).

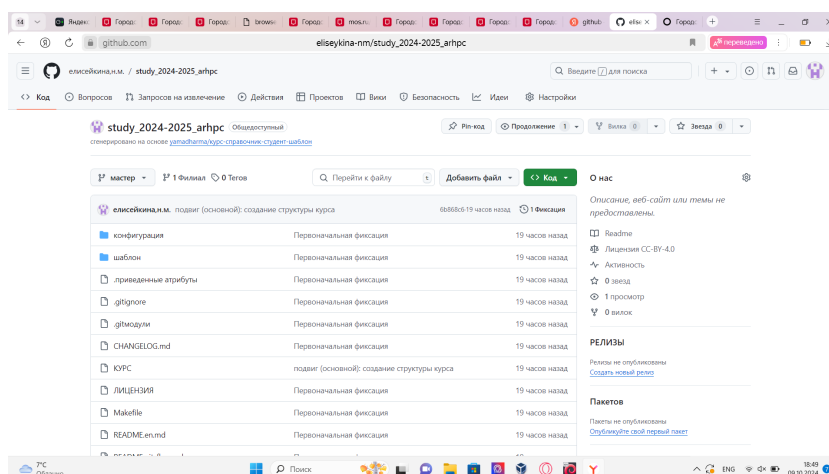


Рис. 4.11: Создание репозитория

11. Перешли в каталог курса (рис. 4.12 Переход в каталог).

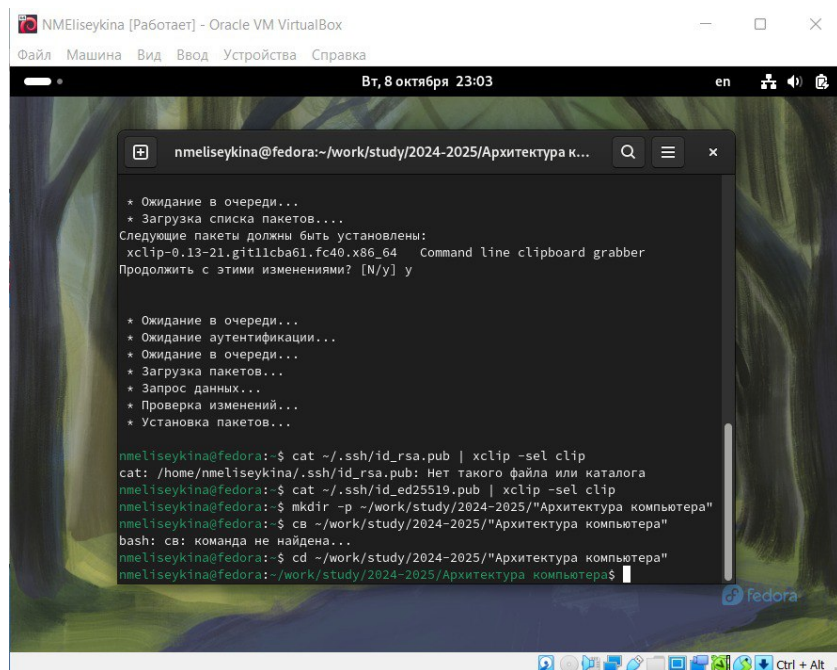
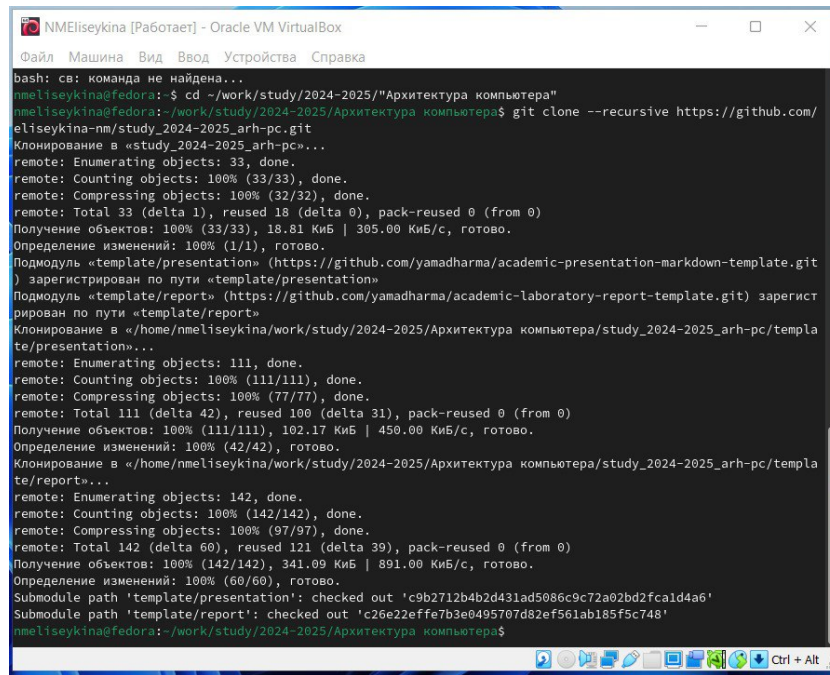


Рис. 4.12: Переход в каталог

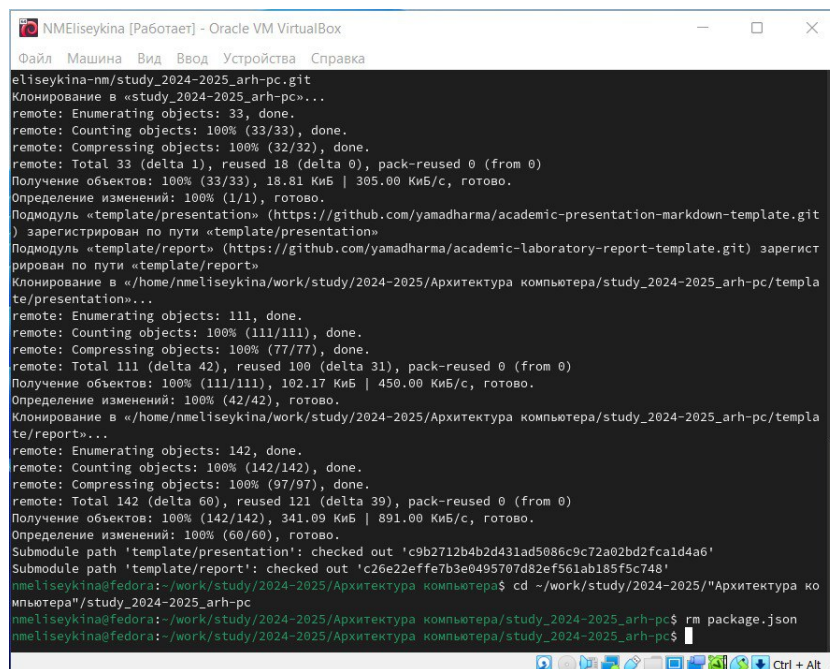
12. Клонировали репозиторий (рис. 4.13 Клонирование репозитория).



```
bash: св: команда не найдена...
nmeliseykina@fedora:~$ cd ~/work/study/2024-2025/Архитектура компьютера
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive https://github.com/nmeliseykina-nm/study_2024-2025_arh-pc.git
Клонирование в «study_2024-2025_arh-pc»...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (33/33), 18.81 КиБ | 305.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/nmeliseykina/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/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 | 450.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/nmeliseykina/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/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 | 891.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера$
```

Рис. 4.13: Клонирование репозитория

13. Удалили лишние файлы (рис. 4.14 Ввод команды для удаления).



```
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера$ cd ~/work/study/2024-2025/Архитектура компьютера
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера$ rm package.json
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера$
```

Рис. 4.14: Ввод команды для удаления

14. Создали каталоги (рис. 4.15 Ввод команды для создания каталогов).

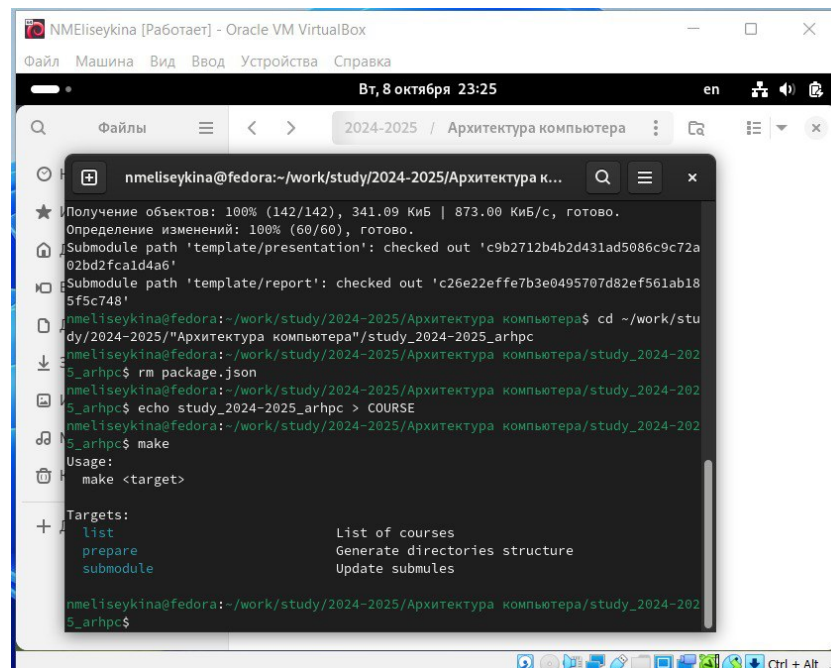


Рис. 4.15: Ввод команды для создания каталогов

15. Отправили файлы на сервер (рис. 4.16 Отправка данных на сервер).

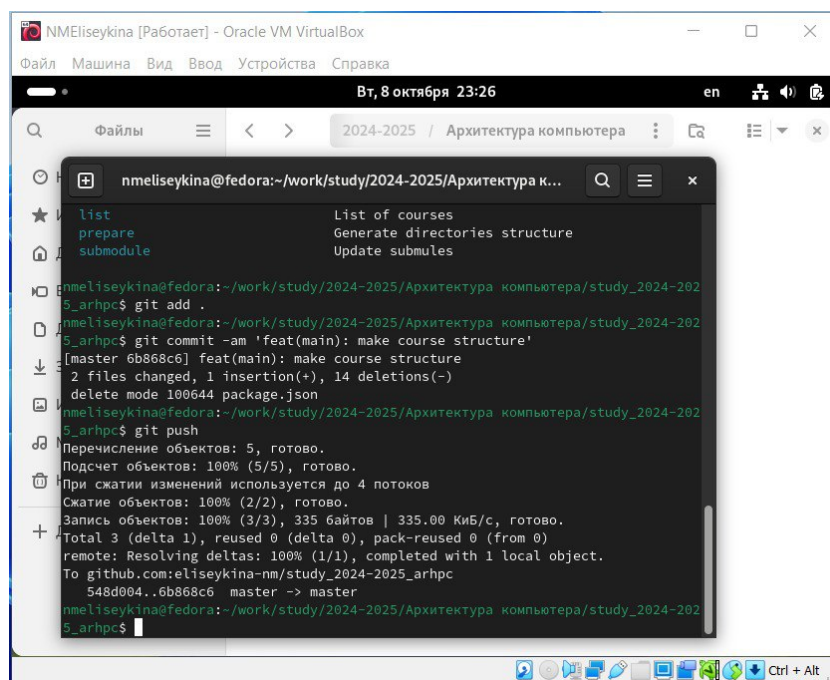


Рис. 4.16: Отправка данных на сервер

16. Проверили правильность создания рабочего пространства в локальном репозитории и на странице github (рис. 4.17 Проверка правильности создания рабочего пространства).

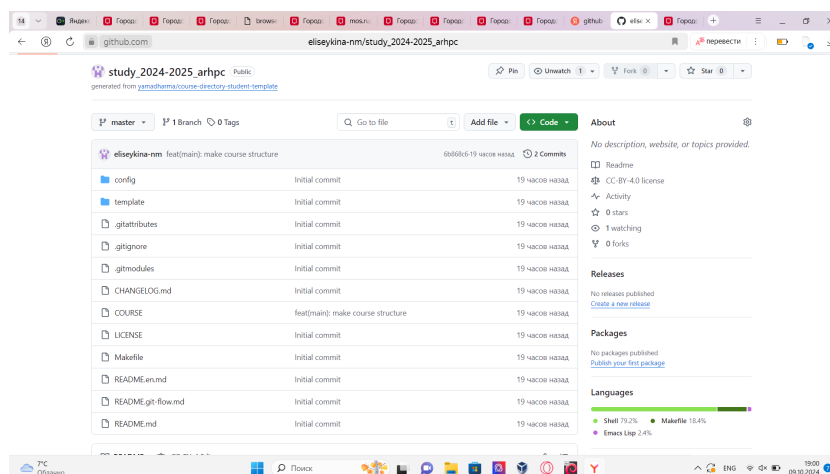


Рис. 4.17: Проверка правильности создания рабочего пространства

Задание для самостоятельной работы

1. Создали каталог для внесения отчета по выполнению лабораторной работы в (labs>lab02>report) (рис. 4.18 Создание каталога).

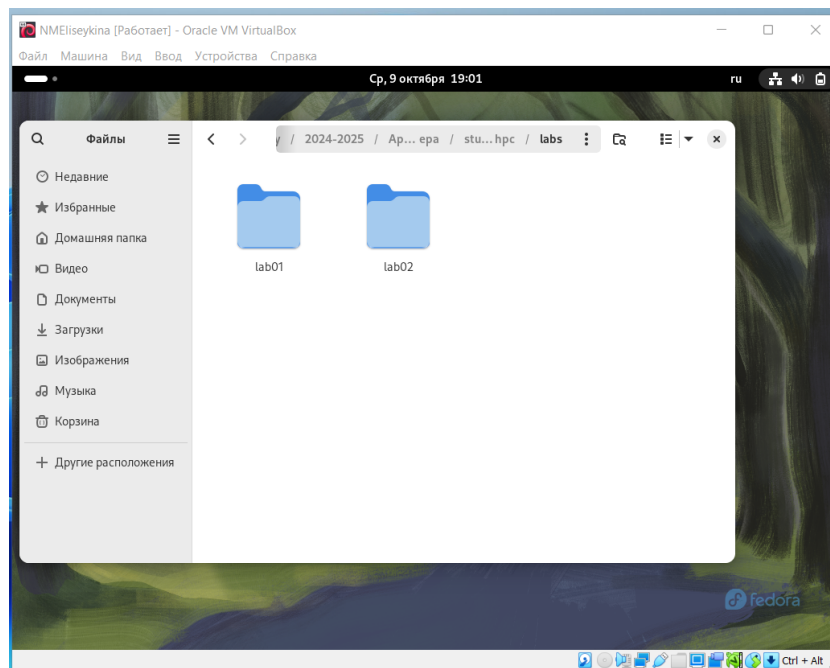
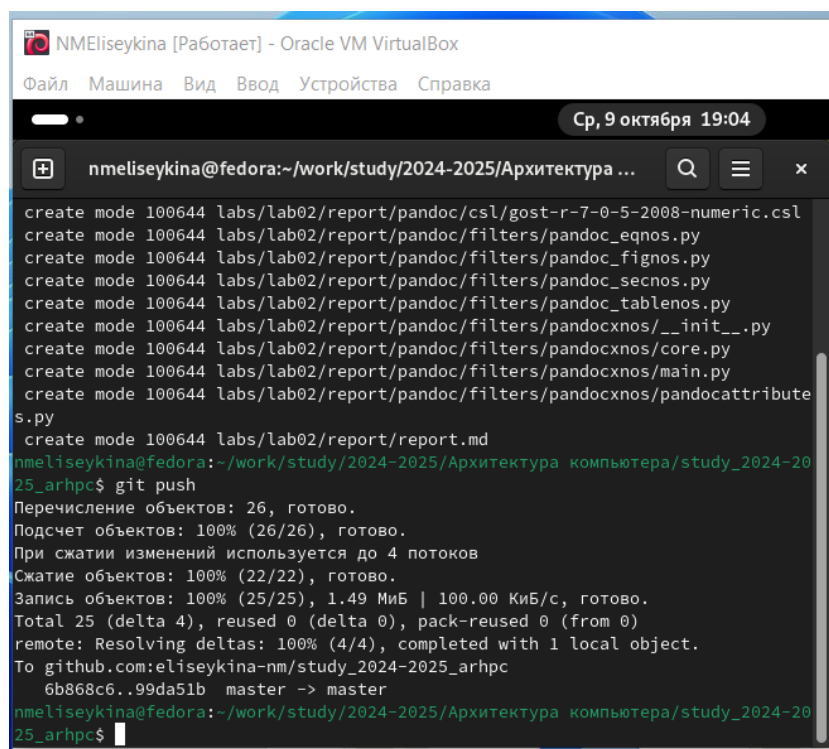


Рис. 4.18: Создание каталога

2. Создали каталог для внесения предыдущих лабораторных работ (рис. 4.19 Создание каталога для предыдущих лабораторных работ).
3. Скопировали отчеты по выполнению предыдущих лабораторных работ в каталоги рабочего пространства.
4. Загрузили файлы на github (рис. 4.20 Загрузка файлов на github).



NMEliseykina [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

Ср, 9 октября 19:04

nmeliseykina@fedora:~/work/study/2024-2025/Архитектура ...

```
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100644 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/pandocattribute
s.py
create mode 100644 labs/lab02/report/report.md
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-20
25_arhpc$ git push
Перечисление объектов: 26, готово.
Подсчет объектов: 100% (26/26), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (22/22), готово.
Запись объектов: 100% (25/25), 1.49 Миб | 100.00 КиБ/с, готово.
Total 25 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:eliseykina-nm/study_2024-2025_arhpc
6b868c6..99da51b master -> master
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-20
25_arhpc$
```

Рис. 4.19: Загрузка файлов на github

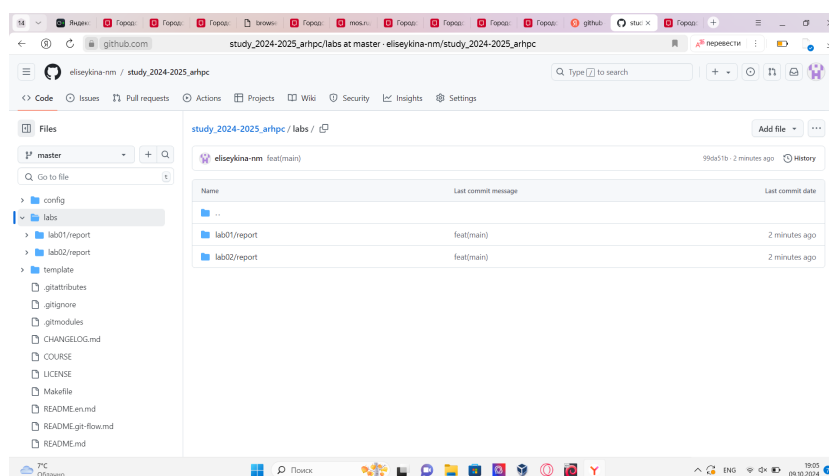


Рис. 4.20: Проверка файлов на github

5 Выводы

Изучили идеологию и применение средств контроля версии. Приобрели практические навыки по работе с git.

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).