РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра фундаментальной информатики и информационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Студент: Терещенкова Маргарита

Группа: НКАбд-04-24

МОСКВА

2024 г.

Содержание

Список иллюстраций

1. Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

2. Теоретическое введение

2.1 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.

2.2. 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.

3. Техническое обеспечение

Лабораторная работа подразумевает выполнение настройки и работы с системой контроля версий Git (https://git-scm.com/). Выполнение работы возможно как в дисплейном классе факультета физико-математических и естественных наук РУДН, так и дома. Описание выполнения работы приведено для дисплейного класса со следующими характеристиками техники:

– Intel Core i3-550 3.2 GHz, 4 GB оперативной памяти, 8 GB свободного места на жёстком диске;

– ОС Linux Gentoo (http://www.gentoo.ru/).

4. Порядок выполнения лабораторной работы

4.1. Настройка github

Создала учётную запись на сайте https://github.com/ и заполнила основные данные.

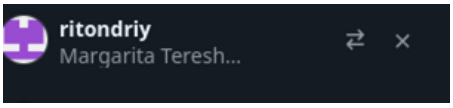

Рисунок 4.1 — Аккаунт GitHub

4.2 Базовая настройка git

Сначала сделала предварительную конфигурацию git.

```
margo@margo-pc:~$ git config --global user.name "<ritondriy>"
margo@margo-pc:~$ git config --global user.email "<ritkasuper18@gmail.com>"
```

Рисунок 4.20 — Предварительная конфигурация git

Настроила utf в выводе сообщений git.

```
margo@margo-pc:~$ git config --global core.quotepath false
```

Рисунок 4.21 — Настройка кодировки

Задала имя «master» для начальной ветки.

```
margo@margo-pc:~$ git config --global init.defaultBranch master
```

Рисунок 4.22 - Создание имени для начальной ветки

Задала параметр autocrlf со значением input.

```
margo@margo-pc:~$ git config --global core.autocrlf input
```

Рисунок 4.23 - Параметр autocrlf

Задала параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость. При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
margo@margo-pc:~$ git config --global core.safecrlf warn
```

Рисунок 4.24 - Параметр safecrlf

4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого вводила команду ssh-keygen -C “Имя Фамилия, work@email”, указывая имя владельца и электронную почту владельца (рис. 4.8). Ключ автоматически сохранится в каталоге ~/.ssh/.

```
margo@margo-pc:~$ ssh-keygen -C "Маргарита Терещенкова <ritkasuper18@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/margo/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/margo/.ssh/id_ed25519
Your public key has been saved in /home/margo/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:/p1/GmXopxafzUekllyhj5uCY2rlVt5WnIWIIFj+sQo Маргарита Терещенкова <ritkas
uper18@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|     o.          |
|    ... .      . |
|      .... . ....|
|       . o. ....+|
|    E   S    .===|
|     . o . . o*B.|
|      . + + ..*=+|
|       . B + *.=*|
|      ..+ o =o+o.|
+----[SHA256]-----+
```

Рисунок 4.30 - Генерация SSH-ключа

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key». Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа.

```
margo@margo-pc:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
```

Рисунок 4.31 — Копирование SSH-ключа

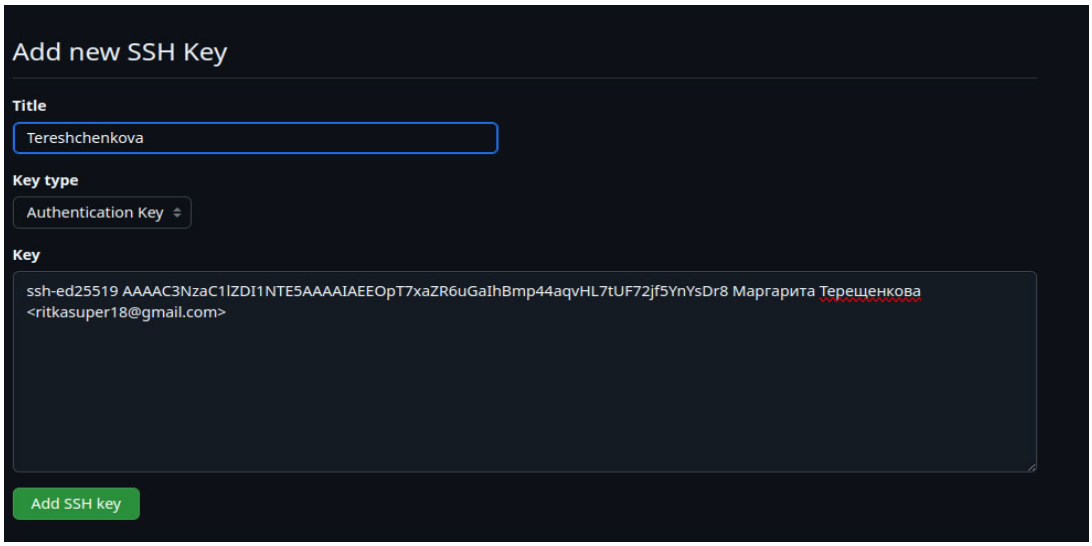

Рисунок 4.32 — Добавление SSH-ключа

4.4. Сознание рабочего пространства и репозитория курса на основе шаблона

Создаю директорию, рабочее пространство, с помощью утилиты mkdir, с помощью ключа -p создаю все директории после домашней ~/work/study/2024-2025/“Архитектура компьютера”

```
margo@margo-pc:~$ mkdir -p ~/work/study/2024-2025/"Архитектура компьютера"
```

Рисунок 4.4 — Создание директории

4.5. Создание репозитория курса на основе шаблона

Перешла на станицу репозитория с шаблоном курса https://github.com/yamadharma/course-directory-student-template. Далее выбираю Use this template.

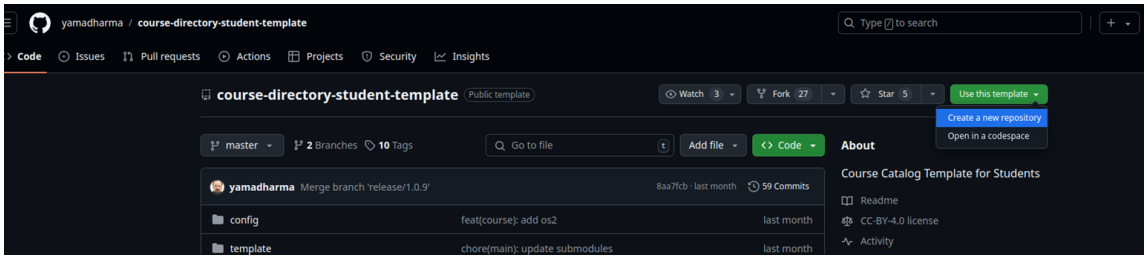

Рисунок 4.50 - Окно Github

В открывшемся окне задаю имя репозитория (Repository name): study_2024-2025_arhpc и создаю репозиторий, нажимаю на кнопку «Create repository from template».

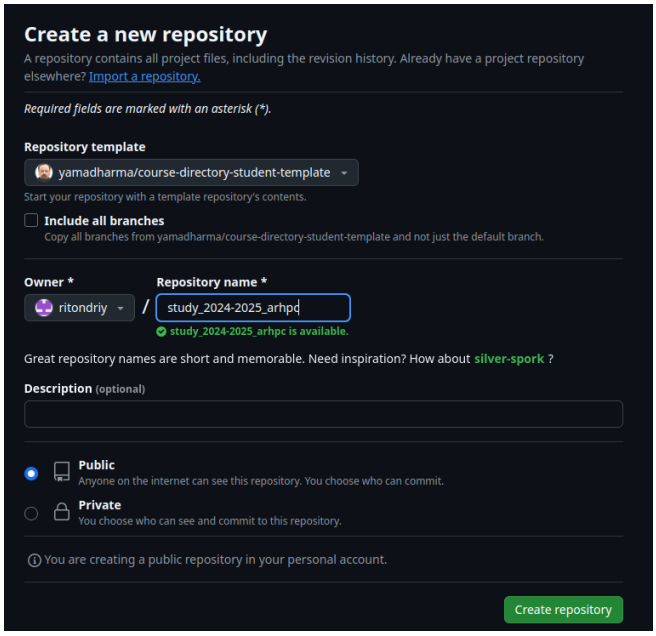

Рисунок 4.51 — Создание репозитория

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study_2024–2025_arh-pc.git arch-pc.

```
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive https://github.com/ritondriy/study_2024-2025_arhpc

Клонирование в «study_2024-2025_arhpc»...
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), 18.82 КиБ | 6.27 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/margo/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/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 | 495.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/margo/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/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 | 992.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
```

Рисунок 4.52 — Клонирование репозитория

4.6. Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd, удаляю лишние файлы с помощью утилиты rm.

```
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера$ cd ~/work/study/2024-2025/Архитектура\ компью
тера/study_2024-2025_arhpc/
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ rm package.json
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$
```

Рисунок 4.60 - Перемещение между директориями и удаление лишних файлов

Создаю каталог, отправляю каталоги с локального репозитория.

```
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ echo study_2024-2025_arhpc > COURSE
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ make
Usage:
  make <target>

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

Рисунок 4.61 - Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit.

```
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git add .
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git commit -am 'feat(main): make cou
rse structure'
[master 5ac4410] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
margo@margo-pc:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git push
```

Рисунок 4.62 - Добавление и сохранение изменений на сервере

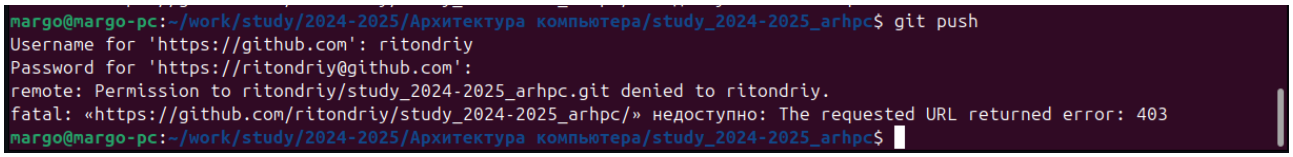

Рисунок 4.63 — Неудача в выгрузке изменений на сервере

Собиралась отправить на сервер, но терминал просит пароль. Я вводила и пароль от git.hub, и токины классические и мелкозерновые. Однако ничего не получается.

5. Задание для самостоятельной работы

1. Перехожу в директорию labs/lab2/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе с помощью утилиты touch.

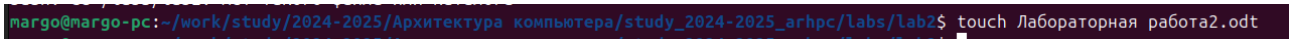

Рисунок 5.1 — Добавление отчета

2. Скопировала отчёт за первую лабораторную работу в папку labs>lab1 аналагично.

3. Не могу загрузить.

6. Контрольные вопросы для самопроверки

1) Системы контроля версий (VCS) — это программное обеспечение, которое помогает отслеживать изменения в файловой системе и эффективно управлять версиями файлов и кода в проекте. Они предназначены для решения следующих задач: Отслеживание изменений: позволяет отслеживать все изменения, сделанные в файлах проекта, включая добавление, удаление, изменение строк кода и текстовых данных. Версионирование: сохраняет изменения в виде версий, что позволяет восстанавливать предыдущие состояния проекта, откатывать изменения и переходить к определённым версиям для просмотра или восстановления кода. Ветвление и слияние (Branching and Merging): позволяет создавать отдельные ветки проекта, где разработчики могут работать независимо, а затем объединять свои изменения в основную ветку. Работа в команде: позволяет нескольким разработчикам работать над одним проектом, автоматически обнаруживая и решая конфликты при слиянии изменений. История изменений: подробно записывает все изменения, включая информацию о коммитах, авторах и времени внесения изменений.
2)Хранилище (репозиторий) — это специальное место, где хранятся файлы и папки проекта. Изменения в этих файлах отслеживаются системой контроля версий (VCS). Рабочая копия — это копия проекта, с которой разработчик работает напрямую. Он вносит изменения в рабочую копию, а затем периодически синхронизирует её с хранилищем. Синхронизация включает отправку изменений, сделанных разработчиком, в хранилище (commit) и актуализацию рабочей копии с последней версией из репозитория (update). История — это последовательность всех изменений, которые были внесены в проект с момента его создания. Она содержит информацию о

том, кто, когда и какие изменения внёс. Таким образом, хранилище служит местом хранения проекта, рабочая копия — инструментом для работы разработчика, а commit и update обеспечивают связь между ними и сохранение истории изменений.
3) Централизованные VCS — это системы контроля версий, в которых репозиторий проекта находится на сервере, доступ к которому осуществляется через клиентское приложение. Примеры централизованных VCS: CVS (Concurrent Versions System) и Subversion (SVN).
Децентрализованные VCS (также называемые распределёнными системами контроля версий, DVCS) хранят копию репозитория у каждого разработчика, работающего с системой. Локальные репозитории периодически синхронизируются с центральным репозиторием. Примеры децентрализованных VCS: Git и Mercurial.
4) При единоличной работе с хранилищем в системе контроля версий (VCS) выполняются следующие действия:
Создание новой папки с датой или пометкой для рабочей версии проекта.
Копирование рабочей версии проекта в новую папку.
Непосредственная работа с рабочей копией проекта.
Периодическая синхронизация рабочей копии с репозиторием путём отправки изменений (commit) и актуализации рабочей копии с последней версией из репозитория (update).
5) Порядок работы с общим хранилищем VCS включает следующие этапы:
Создание репозитория: разработчик создаёт новый репозиторий, используя команды git init, hg init или аналогичные в зависимости от используемой системы контроля версий (например, Git, Mercurial).
Внесение изменений в файлы: разработчик вносит изменения в файлы проекта, например, новые функции, исправления ошибок или другие изменения.
Добавление файлов: разработчик добавляет файлы проекта в список подготовленных для сохранения в репозитории с помощью команды git add, hg add или аналогичной.
Коммит изменений: разработчик фиксирует изменения в файлах, создавая коммит с помощью команды git commit, hg commit или аналогичной. В коммите указывается, какие файлы были изменены, и краткое описание изменений.
Отправка изменений на сервер: разработчик отправляет изменения на сервер с помощью команд git push, hg push или аналогичных.

6) Основные задачи, решаемые инструментальным средством Git:

Возврат к любой предыдущей версии кода.

Просмотр истории изменений.

Параллельная работа над проектом.

Backup кода.

7) git clone — клонирование репозитория в новую директорию. git add — перенос новых и изменённых файлов в проиндексированные. git push — отправка закоммиченных файлов в удалённый репозиторий. git pull — извлечение и загрузка последней информации в локальный репозиторий. git rm — удаление файла из удалённого репозитория.

8) Вот несколько примеров использования локального репозитория:

Создание почтового аккаунта с использованием локального репозитория для хранения данных. Загрузка содержимого сайта с использованием локального репозитория и менеджера файлов.

Настройка доступа к виртуальным папкам с помощью локального репозитория.

Управление базами данных приложений с использованием локального репозитория.

Примеры использования удалённого репозитория:

Разработка программного обеспечения в команде: разработчики могут совместно работать над одним проектом, синхронизируя свои изменения с общим удалённым репозиторием.

Хранение и обмен кодом: разработчики могут делиться своим кодом с другими участниками команды или сообществом, загружая его в удалённый репозиторий.

7. Заключение

Благодаря данной лабораторной работе, я изучила идеологию и применение средств контроля версий. Приобрела практические навыки по работе с системой git.

8. 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://midnightcommander.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 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning- bash-Shell-Programming-Nutshell/dp/0596009658.

6.Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.

7.The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8.Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — 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. — БХВ- Петербург, 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).