Отчёт по лабораторной работе №2

Дисциплина: Архитектура компьютера

Павлова Татьяна Юрьевна

Содержание

# 1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий, а также приобретение практических навыков по работе с системой git.

# 2 Задание

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

# 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.

##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.

# 4 Выполнение лабораторной работы

##Настройка github

Существует несколько доступных серверов репозиториев с возможностью бесплатного размещения данных. Например, http://bitbucket.org/, https://github.com/ и https://gitflic.ru. Для выполнения лабораторных работ предлагается использовать Github. Создайте учётную запись на сайте https://github.com/ и заполните основные данные. (рис. 1).



Рис. 1: (Рис. 1)

##Базовая настройка git

Сначала сделаем предварительную конфигурацию git. Откройте терминал и введите следующие команды, указав имя и email владельца репозитория: git config –global user.name “” git config –global user.email “[work@mail](mailto:work@mail)” (рис. 2).

Рис. 2: (Рис. 2)

Рис. 2: (Рис. 2)

Настроим utf-8 в выводе сообщений git: git config –global core.quotepath false Зададим имя начальной ветки (будем называть её master): git config –global init.defaultBranch master (рис. 3).

Рис. 3: (Рис. 3)

Рис. 3: (Рис. 3)

Параметр autocrlf: git config –global core.autocrlf input Параметр safecrlf: git config –global core.safecrlf warn (рис. 4).

Рис. 4: (Рис. 4)

Рис. 4: (Рис. 4)

##Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C “Имя Фамилия [work@mail](mailto:work@mail)” (рис. 5).

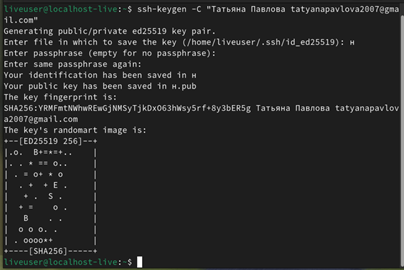


Рис. 5: (Рис. 5)

Ключи сохраняться в каталоге ~/.ssh/. Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайти на сайт http://github.org/ под своей учётной записью и перейти в меню Setting. После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id\_rsa.pub | xclip -sel clip вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title). (рис. 6).

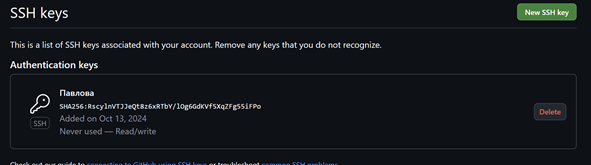


Рис. 6: (Рис. 6)

##Сознание рабочего пространства и репозитория курса на основе шаблона

При выполнении лабораторных работ следует придерживаться структуры рабочего пространства. Рабочее пространство по предмету располагается в следующей иерархии: ~/work/study/ └── / └── / └── / Например, для 2023–2024 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) структура каталогов примет следующий вид: ~/work/study/ └── 2023–2024/ └── Архитектура компьютера/ └── arch-pc/ └── labs/ └── lab01/ └── lab02/ └── lab03/ … • Каталог для лабораторных работ имеет вид labs. • Каталоги для лабораторных работ имеют вид lab, например: lab01, lab02 и т.д. Название проекта на хостинге git имеет вид: study\_\_ Например, для 2023–2024 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) название проекта примет следующий вид: study\_2023–2024\_arch-pc Откройте терминал и создайте каталог для предмета «Архитектура компьютера» (рис. 7).

Рис. 7: (Рис. 7)

Рис. 7: (Рис. 7)

##Сознание репозитория курса на основе шаблона

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейдите на станицу репозитория с шаблоном курса https://github.com/yamadharma/course-directory-student-template. Далее выберите Use this template. (рис. 8).

Рис. 8: (Рис. 8)

Рис. 8: (Рис. 8)

В открывшемся окне задайте имя репозитория (Repository name) study\_2023 2024\_arhpc и создайте репозиторий (кнопка Create repository from template). (рис. 9).

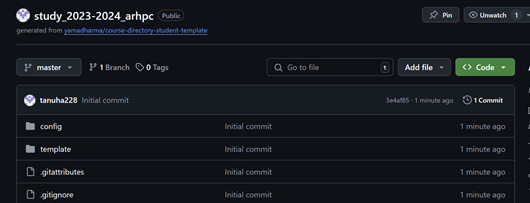


Рис. 9: (Рис. 9)

В открывшемся окне задайте имя репозитория (Repository name) study\_2023–2024\_arhpc и создайте репозиторий (кнопка Create repository from template). Откройте терминал и перейдите в каталог курса: cd ~/work/study/2023–2024/“Архитектура компьютера” клонируйте созданный репозиторий: git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git ↪ arch-pc У меня возникли проблемы с клонированием репозитория с помощью SSH, поэтому я сделала это с помощью HTTPS. (рис. 10).

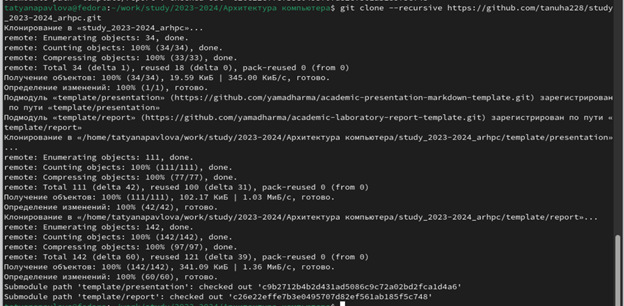


Рис. 10: (Рис. 10)

##Настройка каталога курса

Перейдите в каталог курса: cd ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc Удалите лишние файлы: rm package.json Создайте необходимые каталоги: echo arch-pc > COURSE make (рис. 11).

Рис. 11: (Рис. 11)

Рис. 11: (Рис. 11)

Отправьте файлы на сервер: git add . git commit -am ‘feat(main): make course structure’ (рис. 12).



Рис. 12: (Рис. 12)

git push (рис. 13).

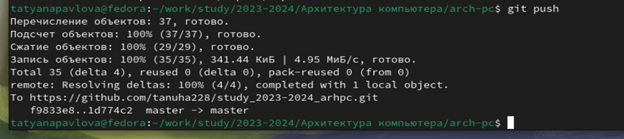


Рис. 13: (Рис. 13)

Проверьте правильность создания иерархии рабочего пространства в локальном репозитории и на странице github. (рис. 14).

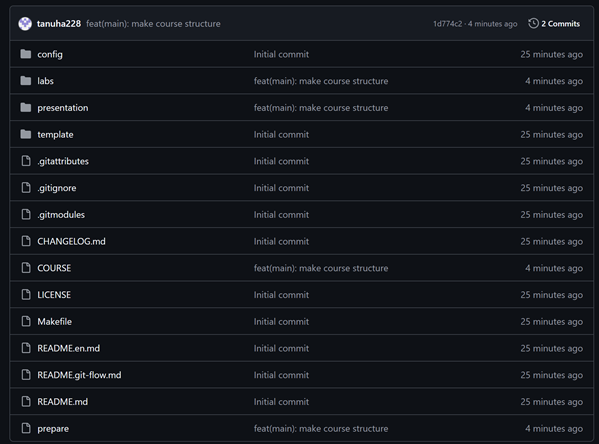


Рис. 14: (Рис. 14)

# 5 Выполнение заданий для самостоятельной работы

Перехожу в подкаталог lab01/report с помощью утилиты cd (рис. 15).

Рис. 15: (Рис. 15)

Рис. 15: (Рис. 15)

Проверяю местонахождение файлов с отчетами по первой лабораторной работе. Они должны быть в подкаталоге домашней директории «Загрузки», для проверки использую команду ls (рис. 16).

Рис. 16: (Рис. 16)

Рис. 16: (Рис. 16)

Копирую первую лабораторную с помощью утилиты cp и проверяю правильность выполнения команды cp с помощью ls (рис. 17).

Рис. 17: (Рис. 17)

Рис. 17: (Рис. 17)

Добавляю с помощью команды git add в коммит файл: Л01\_Павлова\_отчет (рис. 18).

Рис. 18: (Рис. 18)

Рис. 18: (Рис. 18)

Сохраняю изменения на сервере командой git commit -m “…”, поясняя, что добавила файлы (рис. 19).

Рис. 19: (Рис. 19)

Рис. 19: (Рис. 19)

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. 20).

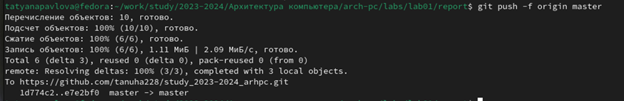


Рис. 20: (Рис. 20)

Проверяю на сайте GitHub правильность выполнения заданий (рис. 21).

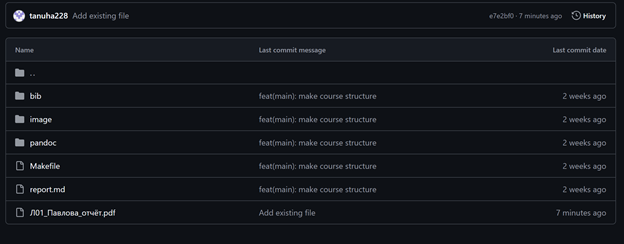


Рис. 21: (Рис. 21)

# 6 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git.

# Список литературы

GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/. 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. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1. 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. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).