Шаблон отчёта по лабораторной работе 2

Архитектура компьютера

Волчкова Елизавета Дмитриевна

Содержание

# Цель работы.

Необходимо понять и изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

# Задание для самостоятельной работы.

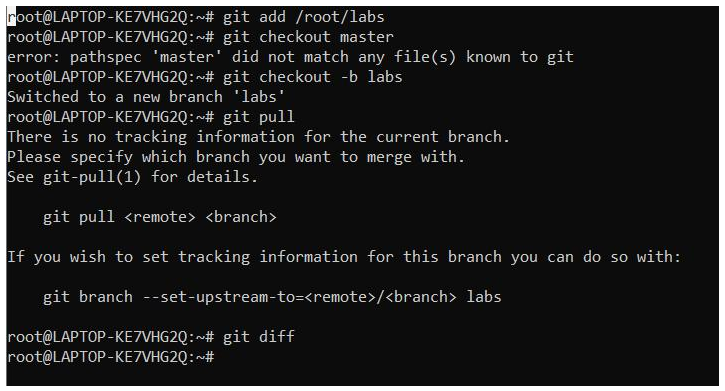
1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github

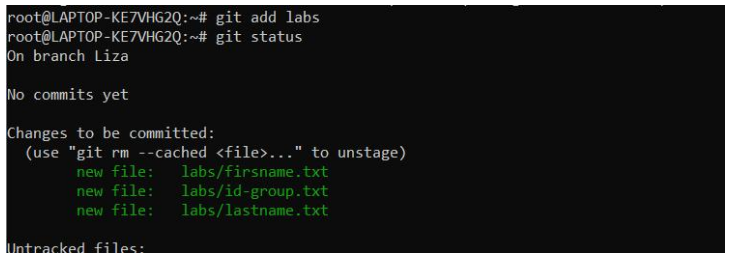
# Теоретическое введение.

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. Демидова А. В. 14 Архитектура ЭВМ 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 6 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

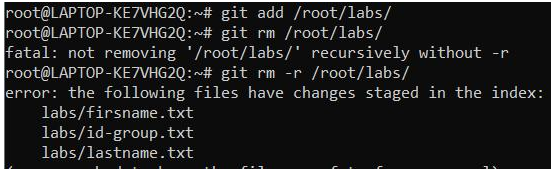
# Выполнение лабораторной работы.

Я начинаю с проверки и получения изменений из центрального репозитория : git checkout master git pull git checkout -b имя\_ветки. Затем вношу изменения в локальном дереве или ветке Далее я разместила их в центральном репозитории. Для этого я проверила, какие файлы изменились к текущему моменту: git status и удалила лишние файлы. Затем посмотрела текст изменений на предмет соответствия правилам ведения чистых коммитов: git diff

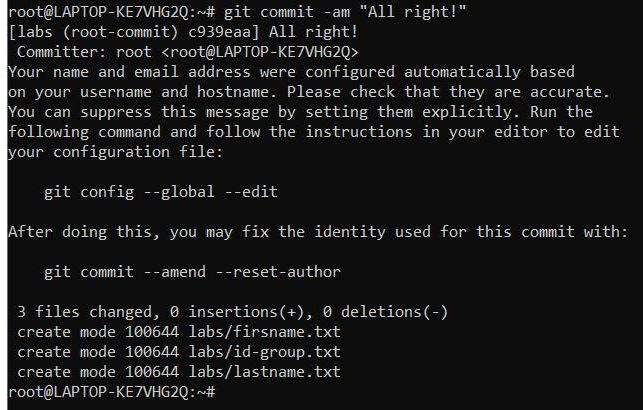
**

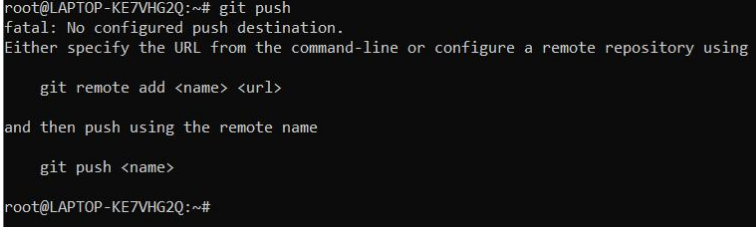
**

Используя команды добавления и/или удаления с нужными опциями: git add имена\_файлов git rm имена\_файлов я пометила файлы, которые не должны попасть в комитет.

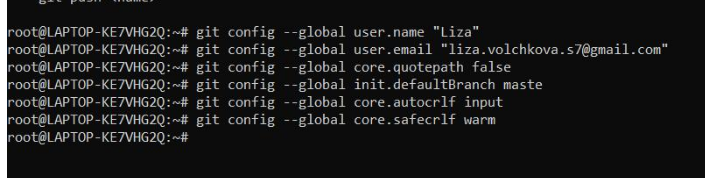
**

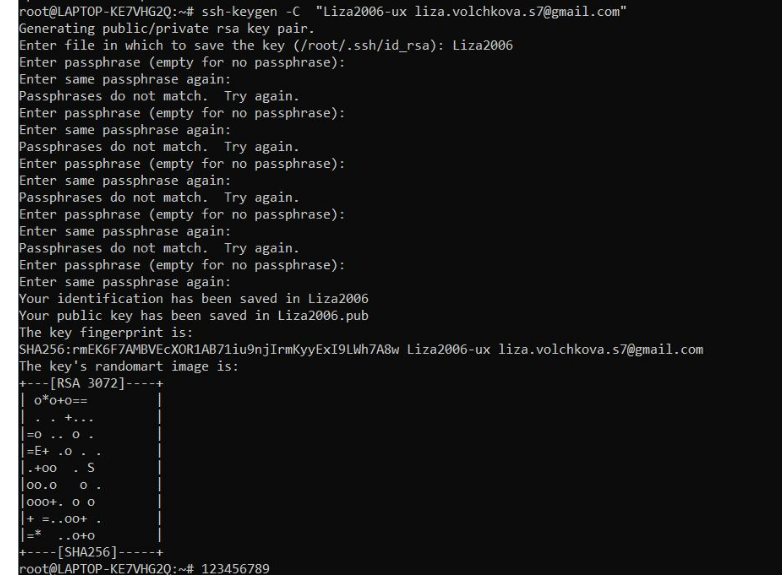
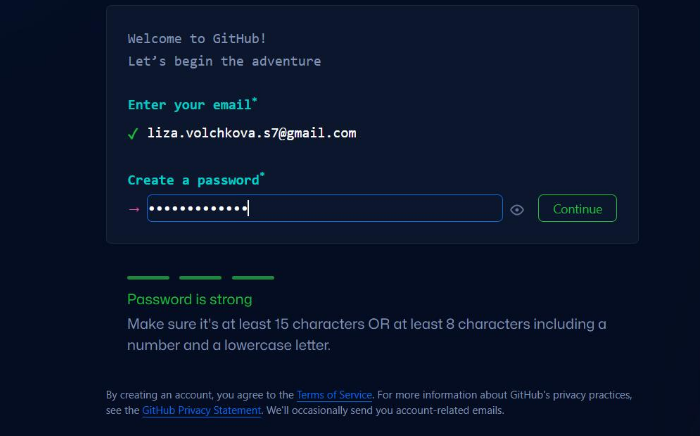
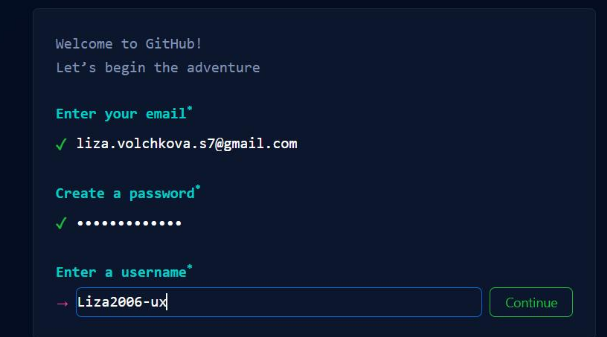
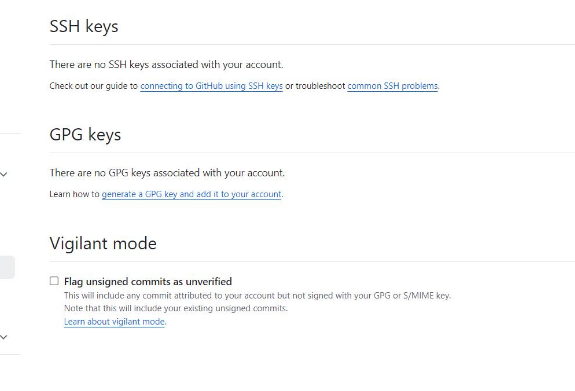
Git add позволяет сохранить все файлы в каталоге . Затем сохранила изменения, поясняя, что было сделано: git commit -am “Some commit message” и отправила в центральный репозиторий: git push origin имя\_ветки или git push.

**

**

Сначала я сделала предварительную конфигурацию git, затем открыла терминал и ввеласледующие команды, указав имя и email владельца репозитория: git config –global user.name “” ” git config –global user.email “[work@mail](about:blank)” Далее настроила utf-8 в выводе сообщений git: git config –global core.quotepath false. Потом задала имя начальной ветки (master): git config –global init.defaultBranch master. Параметр autocrlf: git config –global core.autocrlf input Параметр safecrlf: git config –global core.safecrlf warn

**

Для идентификации пользователя на сервере репозиториев cutythbhjdfkf пару ключей (приватный и открытый): ssh-keygen -C “Liza2006- ux liza.volchkova.s7@gmail.com”.    

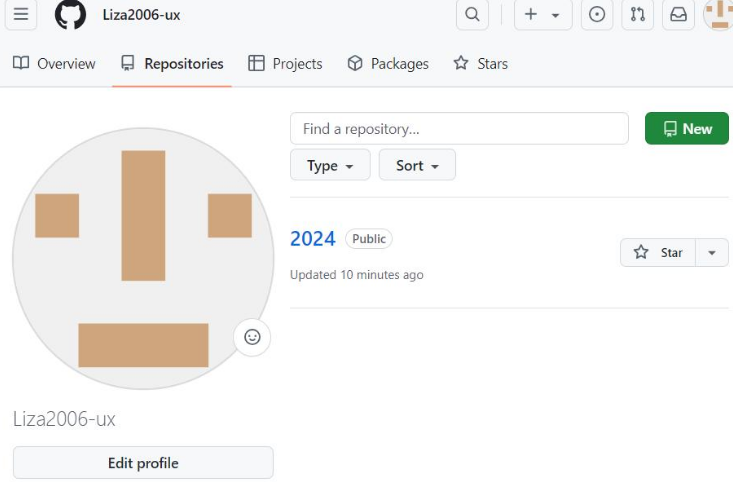
Потом загрузила сгенерённый открытый ключ. Зашла на сайт http: //github.org/ под своей учётной записью и перешла в меню Setting .После этого выбрала в боковом меню SSH and GPG keys и нажала кнопку New SSH key . Скопировав из локальной консоли ключ в буфер обмена cat ~/.ssh/id\_rsa.pub | xclip -sel clip вставила ключ в появившееся на сайте поле и указвала для ключа имя (Title).

Открыла терминал и создала каталог для предмета «Архитектура компьютера»: mkdir -p ~/work/study/2023-2024/“Архитектура компьютера”.

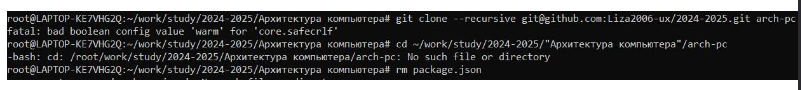
**

Репозиторий на основе шаблона создала через web-интерфейс github. Перейдя на станицу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template.

Далее выбрала Use this template, в открывшемся окне задала имя репозитория (Repository name) study\_2023–2024\_arhpc и создаларепозиторий (кнопка Create repository from template).



Открыла терминал и перешла в каталог курса: cd ~/work/study/2023– 2024/“Архитектура компьютера” клонировала е созданный репозиторий: git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git ↪ arch-pc Ссылку для клонирования скопировала на странице созданного репозитория Code -> SSH:



Потом перешла в каталог курса: cd ~/work/study/2023- 2024/“Архитектура компьютера”/arch-pc, затемудалилалишние файлы: rm package.json



Сoздала необходимые каталоги: echo arch-pc > COURSE make, отпраивла файлы на сервер: git add . git commit -am ‘feat(main): make course structure’ git pus, потом проверила правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

# Вывод.

Целью работы являлось - изучение идеологии и применение средств контроля версий и приобрести практических навыколв по работе с системой git. В процессе работы я использовала разные методы и техники. Так, я разобралась и изучила на практике необходимые команды.

# 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
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,
16. — 874 с. — (Классика Computer Science). 16
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science)