Отчёт по лабораторной работе №2

Дисциплина:Архитектура компьютера и операционные системы

Ванюшкина Татьяна Валерьевна

Table of Contents

# 1 Цель работы

Изучить идеологию и применение средств контроля версий Освоить умения по работе с git

# 2 Теоретическое введение

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.

# 3 Выполнение лабораторной работы

1. Установка программного обеспечения

Устанавливаем git

(рис.1 **¿fig:001?**) ![Установка git](data:application/octet-stream;base64,) {#fig:001}

Устанавливаем gh

(рис.2 **¿fig:002?**) ![Установка gh](data:application/octet-stream;base64,) {#fig:002}

1. Базовая настройка git

Задаем имя и email владельца репозитория

(рис.3 **¿fig:003?**) ![ввод данных о пользователе](data:application/octet-stream;base64,) {#fig:003}

Задаем имя начальной ветки

(рис.4 **¿fig:004?**) ![Имя начальной ветки](data:application/octet-stream;base64,) {#fig:004}

Параметр autocrlf

(рис.5 **¿fig:005?**) ![Параметр autocrlf](data:application/octet-stream;base64,) {#fig:005}

Параметр safecrlf

(рис.6 **¿fig:006?**) ![Параметр safecrlf](data:application/octet-stream;base64,) {#fig:006}

1. Создание ключей

Создаем ключи

(рис.7 **¿fig:007?**) ![Создание ключей](data:application/octet-stream;base64,) {#fig:007}

Генерируем ключ

(рис.9 **¿fig:009?**) ![Генерируем ключ](data:application/octet-stream;base64,) {#fig:009}

Выводим список ключей, копируем отпечаток приватного ключа и копируем генерированный PGP ключ в буфер обмена

(рис.10 **¿fig:010?**) ![Добавление PGP ключа в GitHub](data:application/octet-stream;base64,) {#fig:010}

Перехожу в настройки GitHub (https://github.com/settings/keys), нажмаю на кнопку New GPG key и вставляю полученный ключ в поле ввода.

(рис.11 **¿fig:011?**) ![Добавление PGP ключа в GitHub](data:application/octet-stream;base64,) {#fig:011}

1. Настройка gh

Авторизируемся

(рис.12 **¿fig:012?**) ![Авторизация](data:application/octet-stream;base64,) {#fig:012}

Создаем репозиторий

(рис.13 **¿fig:013?**) ![Создание репозитория](data:application/octet-stream;base64,) {#fig:013}

Удаляем лишние файлы

(рис.14 **¿fig:014?**) ![Удаление лишних файлов](data:application/octet-stream;base64,) {#fig:014}

Создаем необходимые каталоги и добавляем файлы на гитхаб

(рис.15 **¿fig:015?**) ![Создание каталогов](data:application/octet-stream;base64,) {#fig:015}

(рис.16 **¿fig:016?**) ![Репозиторий](data:application/octet-stream;base64,) {#fig:016}

(рис.15 **¿fig:015?**) ![Создание каталогов](data:application/octet-stream;base64,) {#fig:015}

(рис.17 **¿fig:017?**) ![Каталоги](data:application/octet-stream;base64,) {#fig:017}

# 4 Выводы

я изучила идеологию и применение средств контроля версий и освоила умения по работе с git

# Список литературы

Курс: Архитектура компьютеров и операционные системы. Раздел “Операционные системы” (02.03.00, УГСН) (rudn.ru)