Отчет по лабораторной работе №2

Дисциплина: Операционные системы

Беспутин Глеб Антонович

Содержание

1 Цель работы **5**

2 Задание **6**

3 Теоретическое введение **7**

4 Выполнение лабораторной работы **9**
4.1 Установка git 9
4.2 Установка gh 9
4.3 Базовая настройка git 9
4.4 Создаю ключи ssh 10
4.5 Создаем ключи pgp 11
4.6 Настройка github 11
4.7 Добавление PGP ключа в GitHub 11
4.8 Настройка gh 13
4.9 Создаю репозиторий курса на основе шаблона 13
4.10 Настройка каталога курса 14

5 Выводы **16**

6 Контрольные вопросы **17**

Список литературы **19**

Список иллюстраций

4.1 9
4.2 9
4.3 9
4.4 10
4.5 10
4.6 10
4.7 10
4.8 10
4.9 10
4.10 11
4.11 11
4.12 12
4.13 12
4.14 12
4.15 13
4.16 13
4.17 13
4.18 13
4.19 14
4.20 14
4.21 14
4.22 14
4.23 14
4.24 15
4.25 15
4.26 15

Список таблиц

1 Цель работы

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

2 Задание

Создать базовую конфигурацию для работы с git. Создать ключ SSH. Создать ключ PGP. Настроить подписи git. Зарегистрироваться на Github. Создать локальный каталог для выполнения заданий по предмету.

3 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.

4 Выполнение лабораторной работы

4.1 Установка git

Установим git (рис. [4.1]).

Рис. 4.1:

4.2 Установка gh

Установим gh (рис. [4.2]).

Рис. 4.2:

4.3 Базовая настройка git

Зададим имя и email владельца репозитория (рис. [4.3]).

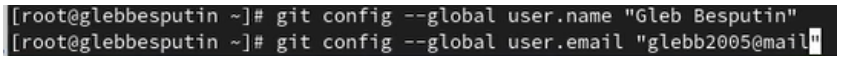

Рис. 4.3:

Настроим utf-8 в выводе сообщений git (рис. [4.4]).

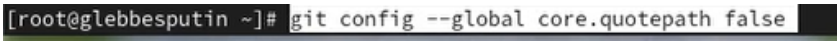

Рис. 4.4:

Зададим имя начальной ветки (будем называть её master) (рис. [4.5]).

Рис. 4.5:

Параметр autocrlf (рис. [4.6]).

Рис. 4.6:

Параметр safecrlf(рис. [4.7]).

Рис. 4.7:

4.4 Создаю ключи ssh

По алгоритму rsa с ключём размером 4096 бит (рис. [4.8]).

Рис. 4.8:

По алгоритму ed25519 (рис. [4.9]).

Рис. 4.9:

4.5 Создаем ключи pgp

Генерируем ключ (рис. [4.10]).

Рис. 4.10:

4.6 Настройка github

Создаю учетную запись(рис. [4.11]).

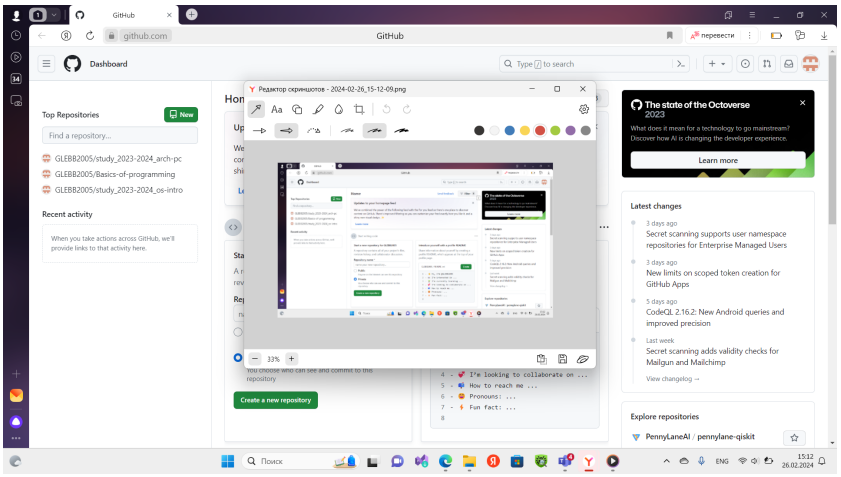

Рис. 4.11:

4.7 Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа(рис. [4.12]).

```
pub   rsa4096 2024-02-25 [SC]
      6292E5BB60AE3A6C7BC2941603052075D0A5E41F
uid                      Gleb Besputin <glebb2005@mail.ru>
sub   rsa4096 2024-02-25 [E]

[root@glebbesputin ~]# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
------------------------
sec   rsa4096/03052075D0A5E41F 2024-02-25 [SC]
      6292E5BB60AE3A6C7BC2941603052075D0A5E41F
uid                 [  абсолютно ] Gleb Besputin <glebb2005@mail.ru>
ssb   rsa4096/5F84F3E76DA0082A 2024-02-25 [E]

[root@glebbesputin ~]#
```

Рис. 4.12:

Копирую свой сгенерированный PGP ключ в буфер обмена (рис. [4.13]).

```
[root@glebbesputin ~]# gpg --armor --export <PGP Fingerprint> | xclip -sel clip
-bash: синтаксическая ошибка рядом с неожиданным маркером «|»
```

Рис. 4.13:

Перехожу в настройки GitHub (https://github.com/settings/keys), нажмаю на кнопку New GPG key и вставляю полученный ключ в поле ввода.(рис. [4.14]).

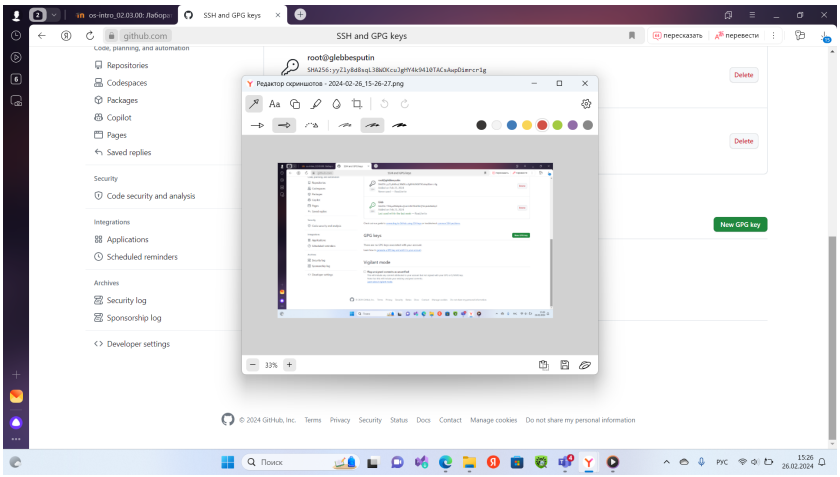

Рис. 4.14:

Используя введёный email, указываю Git применять его при подписи коммитов

4.8 Настройка gh

Авторизуюсь (рис. [4.15]).

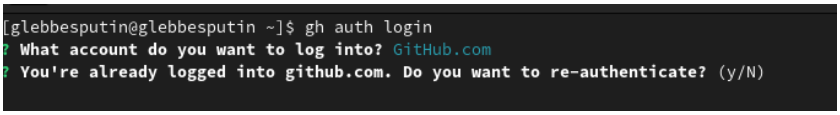

Рис. 4.15:

4.9 Создаю репозиторий курса на основе шаблона

Рис. 4.16:

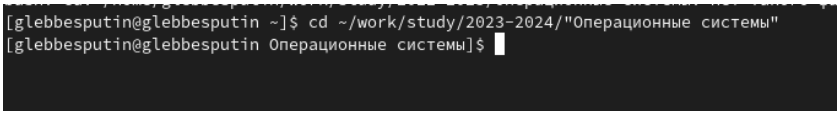

Рис. 4.17:

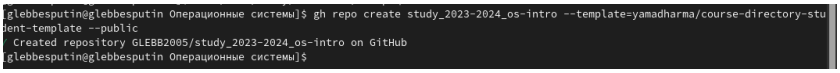

Рис. 4.18:

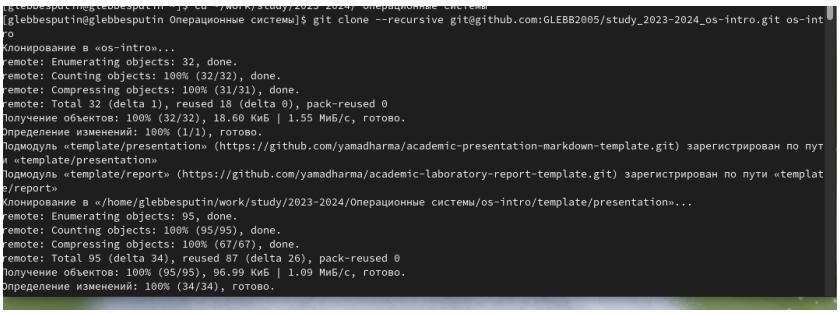

Рис. 4.19:

4.10 Настройка каталога курса

Перехожу в каталог курса (рис. [4.20]).

Рис. 4.20:

Удаляю лишние файлы (рис. [4.21]).

Рис. 4.21:

Создайте необходимые каталоги(рис. [4.22]).

Рис. 4.22:

Рис. 4.23:

Отправьте файлы на сервер(рис. [4.24]).

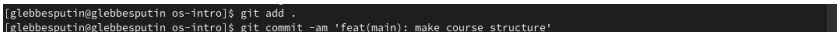

Рис. 4.24:

Рис. 4.25:

Рис. 4.26:

5 Выводы

Создал базовую конфигурацию для работы с git. Создал ключ SSH. Создал ключ PGP. Настроил подписи git. Зарегистрировался на Github. Создал локальный каталог для выполнения заданий по предмету.

6 Контрольные вопросы

Системы контроля версий (VCS) это программное обеспечение, которое позволяет отслеживать изменения в исходном коде и других файлах проекта, сохранять различные версии файлов, а также возвращаться к предыдущим версиям. Основной задачей VCS является упрощение совместной работы над проектом, отслеживание изменений и управление версиями файлов.

Хранилище (repository) - это место, где хранятся все файлы проекта и его история изменений.

Commit - это действие, при котором изменения в файлах проекта фиксируются в репозитории. Коммиты позволяют сохранять изменения и привязывать их к определенному моменту в истории проекта.

История (history) - это список всех изменений проекта, начиная с первоначальной версии. История позволяет отслеживать все изменения, внесенные в проект, и возвращаться к прошлым состояниям.

Рабочая копия (working copy) - это локальная копия файлов из репозитория, с которой вы работаете на компьютере. Рабочая копия позволяет вносить изменения, коммитить их и взаимодействовать с репозиторием.

Централизованные VCS хранят все файлы и историю изменений на центральном сервере, к которому подключаются все пользователи. Пример централизованной VCS - SVN (Subversion). Децентрализованные VCS хранят копии репозитория на компьютерах каждого пользователя, что позволяет им работать независимо от центрального сервера. Пример децентрализованной VCS - Git.

При единоличной работе с хранилищем в VCS можно вносить изменения в

файлы проекта, коммитить их в свой локальный репозиторий и отслеживать историю изменений.

Порядок работы с общим хранилищем VCS включает в себя извлечение последних изменений из центрального репозитория (pull), внесение изменений в рабочую копию, коммит изменений в локальный репозиторий и отправку изменений в центральное хранилище (push).

Git - это инструментальное средство, предназначенное для управления версиями файлов, отслеживания изменений и совместной работы над проектами.

Команды git:

git init - создание нового репозитория git add - добавление файлов в индекс git commit - фиксация изменений в репозитории git push - отправка изменений в удаленный репозиторий git pull - извлечение изменений из удаленного репозитория git branch - создание и управление ветками git merge - объединение веток Примеры использования git при работе с локальным репозиторием: создание нового репозитория, добавление файлов, коммит изменений. Примеры использования git при работе с удалённым репозиторием: отправка изменений на удаленный сервер, извлечение изменений из удаленного репозитория.

Ветви (branches) позволяют отделять различные варианты разработки проекта и работать над ними независимо друг от друга. Ветви могут быть созданы

Список литературы