

Отчет по лабораторной работе №2

Дисциплина: Операционные системы

Оширова Юлия Николаевна, НКАбд-02-22

Содержание

1 Цель работы	5
2 Задание	6
3 Теоретическое введение	7
4 Выполнение лабораторной работы	8
5 Выводы	15
6 Контрольные вопросы для самопроверки	16
Список литературы	20

Список иллюстраций

4.1	Базовая настройка Git	8
4.2	Настройка utf-8 в выводе сообщений git	8
4.3	Задала имя начальной ветки master	8
4.4	Задала параметр autocrlf и safecrlf	9
4.5	Создание SSH ключа	9
4.6	Копирование в буфер обмена	9
4.7	Загрузка сгенерированного SSH ключа	10
4.8	Генерация GPG ключа	11
4.9	Выбираем тип, размер и срок действия	11
4.10	Выводим список ключей и копируем отпечаток приватного ключа	12
4.11	Экспортируем ключ в формате ASCII по его отпечатку	12
4.12	Вставляем полученный ключ в поле ввода	12
4.13	Настройка автоматических подписей коммитов git	13
4.14	Настройка gh	13
4.15	Создание репозитория курса на основе шаблона	14

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

2 Задание

1. Зарегистрироваться на Github;
2. Создать базовую конфигурацию для работы с git;
3. Создать ключ SSH;
4. Создать ключ PGP;
5. Настроить подписи git;
6. Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

В этой лабораторной работе мы познакомимся с системами контроля версий. 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. Существуют классические и распределенные системы контроля версий (РСКВ). Сегодня мы будем работать с распределенной VSC – Git. В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов – они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных. Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах. [1]

4 Выполнение лабораторной работы

№1

Создаем учтную запись на Github и заполняем основные данные.

№2

Сделала предварительную конфигурацию git: открыла терминал и ввела следующие команды, указав имя и email владельца репозитория:

```
git config -g user.name "" git config -g user.email "work@email"
```

Настроила utf-8 в выводе сообщений git: git config -g core.quotepath false

Задала имя начальной ветки master: git config -g init.defaultBranch master

Задала параметр autocrlf: git config -g core.autocrlf input

Задала параметр safecrlf: git config -g core.safecrlf warn

```
ynoshirova@ynoshirova-VirtualBox: ~ git config -g user.name "<joshirova>"  
ynoshirova@ynoshirova-VirtualBox: ~ git config -g user.email "<weiwei04@mail.ru>"  
ynoshirova@ynoshirova-VirtualBox: ~"
```

Рис. 4.1: Базовая настройка Git

```
ynoshirova@ynoshirova-VirtualBox: ~ git config -g core.quotepath false  
ynoshirova@ynoshirova-VirtualBox: ~"
```

Рис. 4.2: Настройка utf-8 в выводе сообщений git

```
ynoshirova@ynoshirova-VirtualBox: ~ git config -g init.defaultBranch master  
ynoshirova@ynoshirova-VirtualBox: ~"
```

Рис. 4.3: Задала имя начальной ветки master

```
ynoshirova@ynoshirova-VirtualBox: $ git config --global init.defaultBranch master
ynoshirova@ynoshirova-VirtualBox: $ git config --global core.autocrlf input
ynoshirova@ynoshirova-VirtualBox: $ git config --global core.safecrlf warn
ynoshirova@ynoshirova-VirtualBox: $
```

Рис. 4.4: Задала параметр autocrlf и safecrlf

№3

Сгенерировала пару ключей – приватный и открытый:

ssh-keygen -C “Имя Фамилия work@mail

```
ynoshirova@ynoshirova-VirtualBox:~$ ssh-keygen -C "joshirova <weiwei04@mail.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ynoshirova/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ynoshirova/.ssh/id_rsa
Your public key has been saved in /home/ynoshirova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:nixlladx86UXZ/u/rGAWJzXXjTCiQYbGNdhsEtz0sP4 joshirova <weiwei04@mail.ru>
The key's randomart image is:
+---[RSA 3072]---+
|   o. @0 . o   |
|   B.=B o o .o|
|   . oo = +o.o*|
|   . . =.oo++|
|   S .o .o..|
|   = o + ..|
|   . + E+ .|
|   . o . . .|
|   ...oo|
+---[SHA256]---+
ynoshirova@ynoshirova-VirtualBox:~$
```

Рис. 4.5: Создание SSH ключа

Скопировала из локальной консоли ключ в буфер обмена:

cat ~/.ssh/id_rsa.pub | xclip -sel clip

```
ynoshirova@ynoshirova-VirtualBox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
ynoshirova@ynoshirova-VirtualBox:~$
```

Рис. 4.6: Копирование в буфер обмена

Загрузила сгенерированный открытый ключ: зашла на сайт <https://github.org/> под своей учетной записью и перешла в меню Settings (настройки), в боковом меню выбрали «SSH and GPG keys» и нажала кнопку «New SSH key», вставила ключ в появившемся на сайте поле и указали имя для ключа – Title:

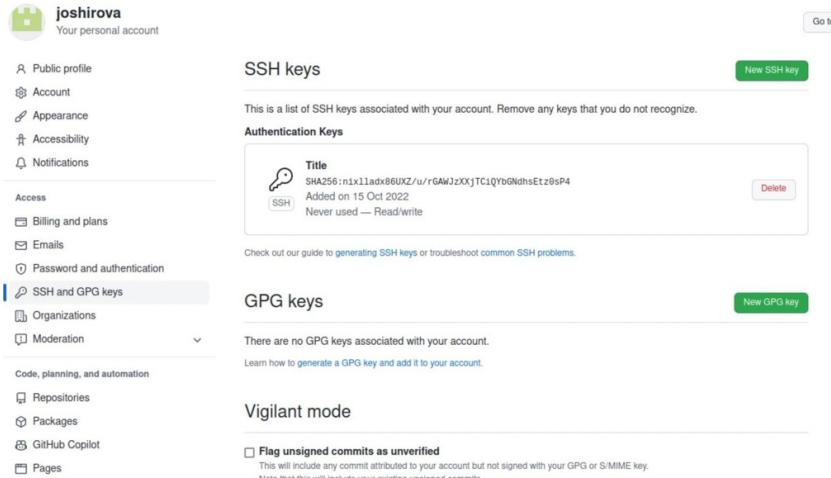


Рис. 4.7: Загрузка сгенерированного SSH ключа

№4

Генерируем ключ

`gpg --full-generate-key`

Из предложенных опций выбираем: тип RSA and RSA; размер 4096; выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).

GPG запросит личную информацию, которая сохранится в ключе: Имя (не менее 5 символов). Адрес электронной почты. При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
ynoshirova@ynoshirova-VirtualBox:/tmp$ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n>  = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) ■
```

Рис. 4.8: Генерация GPG ключа

```
(14) Имеющийся на карте ключ
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n>  = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Yulia
Адрес электронной почты: weiwei04@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Yulia <weiwei04@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
```

Рис. 4.9: Выбираем тип, размер и срок действия

Выводим список ключей и копируем отпечаток приватного ключа:

`gpg --list-secret-keys --keyid-format LONG`

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

Формат строки:

`sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]ID_ключа`

Экспортируем ключ в формате ASCII по его отпечатку:

```
gpg --armor --export
```

```
ynoshirova@ynoshirova-VirtualBox:/tmp$ gpg --list-secret-keys --keyid-format LONG  
G  
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  
/home/ynoshirova/.gnupg/pubring.kbx  
-----  
sec rsa4096/FBB4B5B31DD8B103 2023-02-16 [SC]  
      526D16CE375F9C46316B9AAEFBB4B5B31DD8B103  
uid          [ absolute ] Yulia <weiwei04@mail.ru>  
ssb rsa4096/FBB260A85CB2DBE3 2023-02-16 [E]
```

Рис. 4.10: Выводим список ключей и копируем отпечаток приватного ключа

```
ynoshirova@ynoshirova-VirtualBox:/tmp$ gpg --armor --export FBB4B5B31DD8B103 | x  
clip -sel clip  
ynoshirova@ynoshirova-VirtualBox:/tmp$
```

Рис. 4.11: Экспортируем ключ в формате ASCII по его отпечатку

Добавление PGP ключа в GitHub

Копируем ключ и добавляем его в настройках профиля на GitHub (или GitLab).

Скопируйте ваш сгенерированный PGP ключ в буфер обмена:

```
gpg --armor --export | xclip -sel clip
```

Перейдите в настройки GitHub (<https://github.com/settings/keys>), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода.

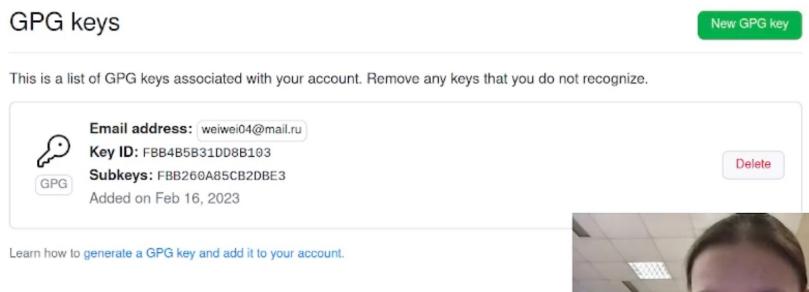


Рис. 4.12: Вставляем полученный ключ в поле ввода

№5

Настройка автоматических подписей коммитов git

Используя введённый email, укажите Git применять его при подписи коммитов:
git config --global user.signingkey git config --global commit.gpgsign true git config
--global gpg.program \$(which gpg2)

```
ynoshirova@ynoshirova-VirtualBox:/tmp$ git config --global user.signingkey FBB4B  
5B31DD8B103  
ynoshirova@ynoshirova-VirtualBox:/tmp$ git config --global commit.gpgsign true  
ynoshirova@ynoshirova-VirtualBox:/tmp$ git config --global gpg.program $(which g  
pg2)  
ynoshirova@ynoshirova-VirtualBox:/tmp$
```

Рис. 4.13: Настройка автоматических подписей коммитов git

Настройка gh

Для начала необходимо авторизоваться

```
gh auth login
```

Утилита задаст несколько наводящих вопросов. Авторизоваться можно через броузер.

```
ynoshirova@ynoshirova-VirtualBox:/tmp$ gh auth login  
? What account do you want to log into? GitHub.com  
? What is your preferred protocol for Git operations? SSH  
? Generate a new SSH key to add to your GitHub account? No  
? How would you like to authenticate GitHub CLI? Login with a web browser  
  
! First copy your one-time code: 9E64-1BDD  
Press Enter to open github.com in your browser... █
```

Рис. 4.14: Настройка gh

№6

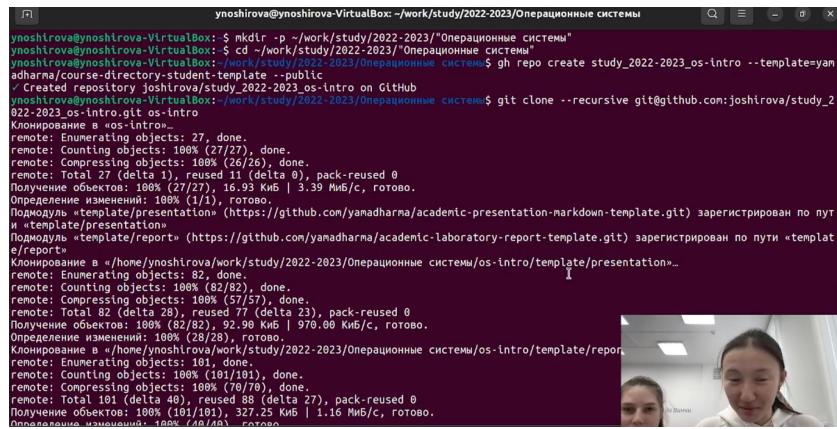
Создание репозитория курса на основе шаблона

Необходимо создать шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы).

Например, для 2022–2023 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

```
mkdir -p ~/work/study/2022-2023/"Операционные системы"  
cd ~/work/study/2022-2023/"Операционные системы"  
gh repo create study_2022-2023_os-intro --template=yamadharma/course-  
directory-student-template --public
```

```
git clone --recursive git@github.com:<owner>/study_2022-2023_os-intro.git os-intro
```



```
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы
ynoshirova@ynoshirova-VirtualBox: $ mkdir -p ~/work/study/2022-2023/"Операционные системы"
ynoshirova@ynoshirova-VirtualBox: $ cd ~/work/study/2022-2023/"Операционные системы"
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы$ gh repo create study_2022-2023_os-intro --template=yan-adharma/course-directory-student-template --public
✓ Created repository joshirova/study_2022-2023_os-intro on GitHub
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы$ git clone --recursive git@github.com:joshirova/study_2022-2023_os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
remote: Получение объектов: 100% (27/27), 16.93 Кб | 3.39 Мб/с, готово.
Определение изменений: 100% (1/1), готово.
Подключен шаблон «template/presentation» (https://github.com/yanadharma/academic-presentation-markdown-template.git) зарегистрирован по пути и «template/presentation»
Подключен шаблон «report» (https://github.com/yanadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в ~/home/ynoshirova/work/study/2022-2023/Операционные системы/os-intro/template/presentation...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 57 (delta 27), pack-reused 0
remote: Получение объектов: 100% (82/82), 63.99 Кб | 970.00 Кб/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в ~/home/ynoshirova/work/study/2022-2023/Операционные системы/os-intro/template/report...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
remote: Получение объектов: 100% (101/101), 327.25 Кб | 1.16 Мб/с, готово.
Определение изменений: 100% (101/101), готово.
```

Рис. 4.15: Создание репозитория курса на основе шаблона

Настройка каталога курса

Перейдите в каталог курса: `cd ~/work/study/2022-2023/“Операционные системы”/os-intro`

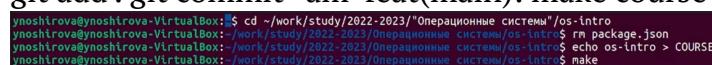
Удалите лишние файлы: `rm package.json`

Создайте необходимые каталоги:

`echo os-intro > COURSE` `make`

Отправьте файлы на сервер:

`git add .` `git commit -am ‘feat(main): make course structure’` `git push`



```
ynoshirova@ynoshirova-VirtualBox: $ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы/os-intro$ rm package.json
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы/os-intro$ echo os-intro > COURSE
ynoshirova@ynoshirova-VirtualBox: ~/work/study/2022-2023/Операционные системы/os-intro$ make
```

5 Выводы

Изучила идеологию и применение средств контроля версий. Приобрели практические навыки по работе с системой git.

6 Контрольные вопросы для самопроверки

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control System, VCS) применяются для:
 - Хранение полной истории изменений
 - причин всех производимых изменений
 - Откат изменений, если что-то пошло не так
 - Поиск причины и ответственного за появления ошибок в программе
 - Совместная работа группы над одним проектом
 - Возможность изменять код, не мешая работе других пользователей
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): • Одно основное хранилище всего проекта • Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно Децентрализованные VCS (Git; Mercurial; Bazaar): • У каждого пользователя свой вариант (возможно не один) репозитория • Присутствует возможность добавлять и забирать изменения из любого репозитория [2] В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.
4. Опишите действия с VCS при единоличной работе с хранилищем. Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
5. Опишите порядок работы с общим хранилищем VCS. 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.
6. Каковы основные задачи, решаемые инструментальным средством git?
Первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам git. Наиболее часто

используемые команды git:

- создание основного дерева репозитория: git init
- получение обновлений (изменений) текущего дерева из центрального репозитория: git pull
- отправка всех произведённых изменений локального дерева в центральный репозиторий: git push
- просмотр списка изменённых файлов в текущей директории: git status
- просмотр текущих изменений: git diff
- сохранение текущих изменений:
 - добавить все изменённые и/или созданные файлы и/или каталоги: git add
 - добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена_файлов
- удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов
- сохранение добавленных изменений:
 - сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’
 - сохранить добавленные изменения с внесением комментария через встроенный редактор git commit
- создание новой ветки, базирующейся на текущей: git checkout -b имя_ветки
- переключение на некоторую ветку: git checkout имя_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
- отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки
- слияние ветки с текущим деревом: git merge –no-ff имя_ветки
- удаление ветки:
 - удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки
 - принудительное удаление локальной ветки: git branch -D имя_ветки
 - удаление ветки с центрального репозитория: git push origin :имя_ветки

8. Приведите примеры использования при работе с локальным и удалённым репозиториями. git push –all (push origin master/любой branch)
9. Что такое и зачем могут быть нужны ветви (branches)? Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть главная

ветка (master), или ствол (trunk). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

10. Как и зачем можно игнорировать некоторые файлы при commit? Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

Список литературы

1. О системе контроля версий. Электронный ресурс. 2016
2. Евгений Г. Системы контроля версий. Электронный ресурс.