

Лабораторная работа 3

Дисциплина операционные системы

Чичкина Ольга Константиновна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	12
Контрольные вопросы	13

Список иллюстраций

0.1	установка программного обеспечения git-flow	7
0.2	базовая настройка git	8
0.3	создание ключей ssh	8
0.4	Генерация ключей pgr	9
0.5	добавление ключа pgr к github	9
0.6	подписи коммитов	10
0.7	настройку gh	10
0.8	создание шаблона рабочего пространства	11

Список таблиц

Цель работы

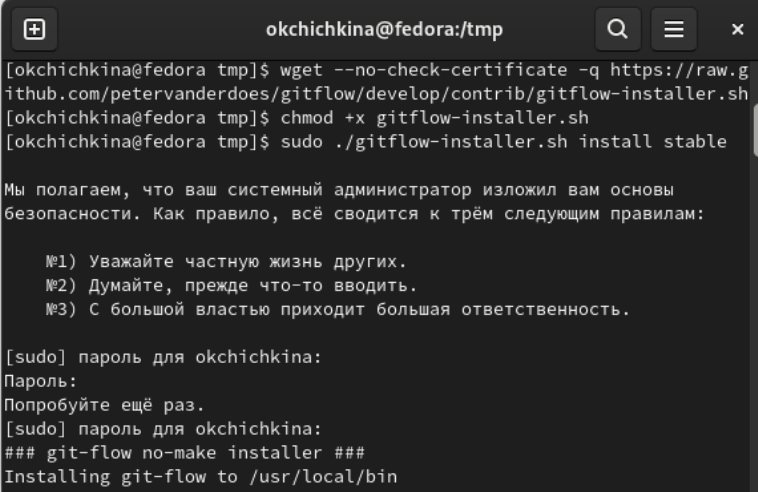
Целью работы является изучение идеологии и применение средств контроля версий. Освоение умения по работе с git.

Задание

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

Выполнение лабораторной работы

Создаем учетную запись на <https://github.com> и заполняем основные данные . Устанавливаем программное обеспечение git-flow , используя команды «wget», «chmod», и «sudo». (рис. [-@fig:001])



```
okchichkina@fedora:/tmp
[okchichkina@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[okchichkina@fedora tmp]$ chmod +x gitflow-installer.sh
[okchichkina@fedora tmp]$ sudo ./gitflow-installer.sh install stable

Мы полагаем, что ваш системный администратор изложил вам основы безопасности. Как правило, всё сводится к трём следующим правилам:

№1) Уважайте частную жизнь других.
№2) Думайте, прежде что-то вводить.
№3) С большой властью приходит большая ответственность.

[sudo] пароль для okchichkina:
Пароль:
Попробуйте ещё раз.
[sudo] пароль для okchichkina:
### git-flow no-make installer ###
Installing git-flow to /usr/local/bin
```

Рис. 0.1: установка программного обеспечения git-flow

Установка gh. Переходим к базовой настройке git , для этого задаем имя и email владельца репозитория, настраиваем utf-8 в выводе сообщений git, а также верификацию и подписание коммитов git, задаем имя начальной ветки (master), устанавливаем параметры autocrlf и safecrlf. (рис. [-@fig:002])

```
okchichkina@fedora:/tmp
gh-2.7.0-1.fc35.x86_64

Выполнено!
[okchichkina@fedora tmp]$ git config --global user.name "ok-chicha"
[okchichkina@fedora tmp]$ git config --global user.email "olka.chichkina@gmail.com"
[okchichkina@fedora tmp]$ git config --global core.quotepath false
[okchichkina@fedora tmp]$ git config --global init.defaultBranch master
[okchichkina@fedora tmp]$ git config --global core.autocrlf input
[okchichkina@fedora tmp]$ git config --global core.safecrlf warn
[okchichkina@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/okchichkina/.ssh/id_rsa):
Created directory '/home/okchichkina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/okchichkina/.ssh/id_rsa
Your public key has been saved in /home/okchichkina/.ssh/id_rsa.pub
The key fingerprint is:
```

Рис. 0.2: базовая настройка git

Создаем ключи ssh по алгоритму rsa с ключём размером 4096 бит .(рис. [-@fig:003])

```
okchichkina@fedora:/tmp

[okchichkina@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/okchichkina/.ssh/id_rsa):
Created directory '/home/okchichkina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/okchichkina/.ssh/id_rsa
Your public key has been saved in /home/okchichkina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zs/iPJqSX9JfnkaQLnLBrtTWs/rBNXuLwfm6gD9Z7U okchichkina@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|          +oo|
|         .oBo|
|        .=ooo|
|       .o . + .|
|      S . = o=.|
|     + .o.oo+oo|
|    .. = .oo =Eo|
|   o =o+ += o |
|  o+ooo+o=    |
+---[SHA256]-----+
[okchichkina@fedora tmp]$ gpg --full-generate-key
```

Рис. 0.3: создание ключей ssh

Генерируем ключи pgr командой «gpg --full-generate-key» и из предложенных опций выбираем: - тип RSA and RSA; - размер 4096; - выберите срок действия - 0 (срок действия не истекает никогда). - Имя (okchichkina). - Адрес электронной почты(olka.chichkina@gmail.com) (рис. [-@fig:004])


```
okchichkina@fedora:tmp

[okchichkina@fedora tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; 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.

gpg: создан каталог '/home/okchichkina/.gnupg'
gpg: создан щит с ключами '/home/okchichkina/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
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
```

Рис. 0.4: Генерация ключей gpg

Для добавления ключа gpg в github нужно вывести список ключей и скопировать отпечаток приватного ключа (86E5C166DD4341D1). Далее мы вставляем полученный отпечаток в команду для копирования ключа gpg, и получаем команду вида «gpg –armor –export 86E5C166DD4341D1 ». после этого мы подключаем полученный ключ на github. (рис. [-@fig:005])

```
Обзор Терминал 17.22 апреля 16:22 en

okchichkina@fedora:tmp

открытый и секретный ключи созданы и подписаны.

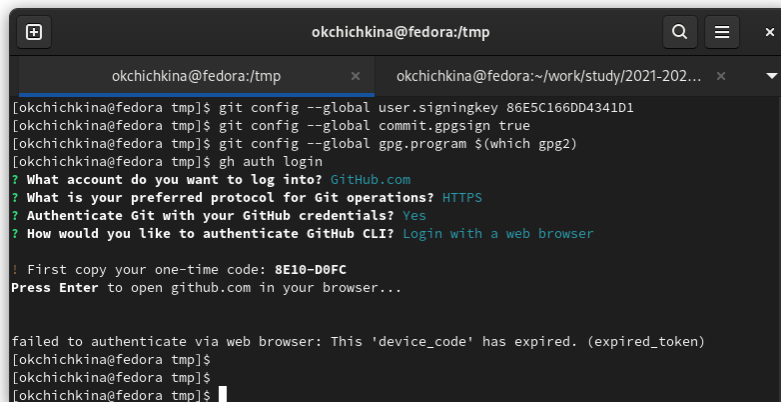
pub  rsa4096 2022-04-22 [SC]
     EC8D1E252EF158F9B388C286E5C166D04341D1
uid   ok-chicha.colka.chichkina@gmail.com
sub   rsa4096 2022-04-22 [E]

[okchichkina@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: PGP
gpg: глубина: 0 достижима: 1 подписанных: 0 доверия: 0-, 0a, 0n, 0f, 1u
/home/okchichkina/.gnupg/pubring.kbx
-----BEGIN PGP PRIVATE KEY BLOCK-----
sec  rsa4096/86E5C166D04341D1 2022-04-22 [SC]
     EC8D1E252EF158F9B388C286E5C166D04341D1
uid   [ aScometmo ] ok-chicha.colka.chichkina@gmail.com
sub   rsa4096/6C8E2A50F466D5F 2022-04-22 [E]
-----BEGIN PGP PRIVATE KEY BLOCK-----
hQTNBQ3In/ABEAc2lXB8tk+uSoc2ZlXj+FxFutv84BYCOAnB1Sk3hv1Mb5EDapt
nZ/Way2sgjOKM2g8HICqLhTEI1CRV2kxRU0gDmk+ztWf+aPRgV81OKfPa
s/KQM2/3FvV13IC/0hG2Wj1K/pa5d6karrzY80FwBzuzEYwP+L4IRhD01w
tG0hK00ayz0hm1lFaywA7k8a8Qm8+SC13hV/Tc5s1Vp4T840uWd6CpF
n4j3/xMroBdZD2NPteHk9L6FFPKa8B851v3qBv5UTxj1Juko2Qn8E4w/tKuUv
h8kw8ez0R0S4jQCfpZCpVv/vkCatm+zdva+5ERLI08UX1X3cacQ+ssAF1Dd
u0PUEt11018t8nuV853t6Pw30NM8ATVqJ3E3jw0bHkcm1j3a10b0Tc5
```

Рис. 0.5: добавление ключа gpg к github

Используя email, указываем git применять его при подписи коммитов (рис. [-

@fig:006))

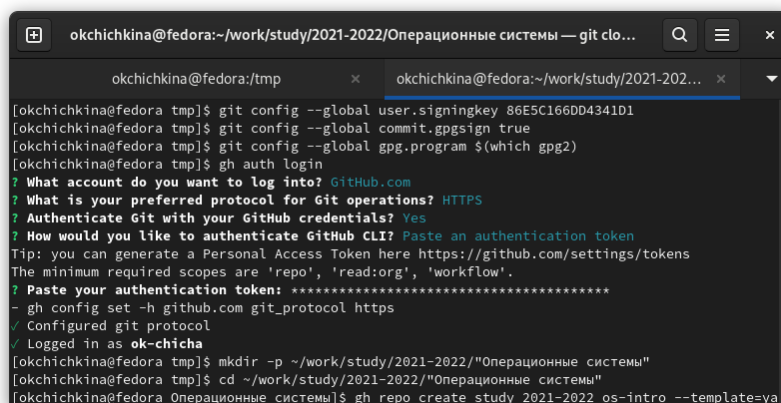


```
okchichkina@fedora:tmp
[okchichkina@fedora tmp]$ git config --global user.signingkey 86E5C166DD4341D1
[okchichkina@fedora tmp]$ git config --global commit.gpgsign true
[okchichkina@fedora tmp]$ git config --global gpg.program $(which gpg2)
[okchichkina@fedora tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser
! First copy your one-time code: 8E10-D0FC
Press Enter to open github.com in your browser...

failed to authenticate via web browser: This 'device_code' has expired. (expired_token)
[okchichkina@fedora tmp]$
[okchichkina@fedora tmp]$
[okchichkina@fedora tmp]$
```

Рис. 0.6: подписи коммитов

Начинаем выполнять настройку gh, для этого авторизовываемся и отвечаем на вопросы утилиты .(рис. [-@fig:007])



```
okchichkina@fedora:~/work/study/2021-2022/Операционные системы — git clo...
[okchichkina@fedora tmp]$ git config --global user.signingkey 86E5C166DD4341D1
[okchichkina@fedora tmp]$ git config --global commit.gpgsign true
[okchichkina@fedora tmp]$ git config --global gpg.program $(which gpg2)
[okchichkina@fedora tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as ok-chicha
[okchichkina@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[okchichkina@fedora tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
[okchichkina@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=ya
```

Рис. 0.7: настройку gh

Переходим к созданию шаблона рабочего пространства и настройке каталога курса. (рис. [-@fig:008])

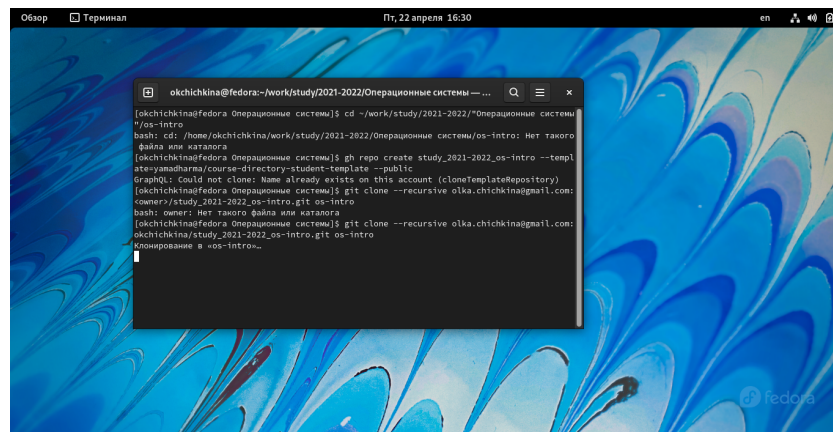


Рис. 0.8: создание шаблона рабочего пространства

Выводы

Изучила идеологию и научилась применять средства контроля версий..

Контрольные вопросы

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
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. 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.
4. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений `git config --global core.quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`
5. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняться в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.
6. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Основные команды 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 изменения: `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 стекущим 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. Использование `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit -am 'Новый файл'`
9. Проблемы, которые решают ветки `git`:
 - нужно постоянно создавать архивы с рабочим кодом
 - сложно “переключаться” между архивами
 - сложно перетаскивать изменения между архивами
 - легко что-то напутать или потерять
10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуются добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавле-

нии в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c » .gitignore` `curl -L -s https://www.gitignore.io/api/c++ » .gitignore`