Шаблон отчёта по лабораторной работе номер 2

Дисциплина: Операционные системы

Крестененко Полина Александровна

Содержание

Список таблиц

Список иллюстраций

1 Цель работы

Изучить идеологию и применение средств контроля версий.

2 Задание

Написать отчет по 2 лабораторной работе в формате Makefile. Сохранить отчет в трех разных форматах.

3 Выполнение лабораторной работы

1) Настройка git • Создаю учетную запись на https://github.com. (рис. -fig. 3.1)

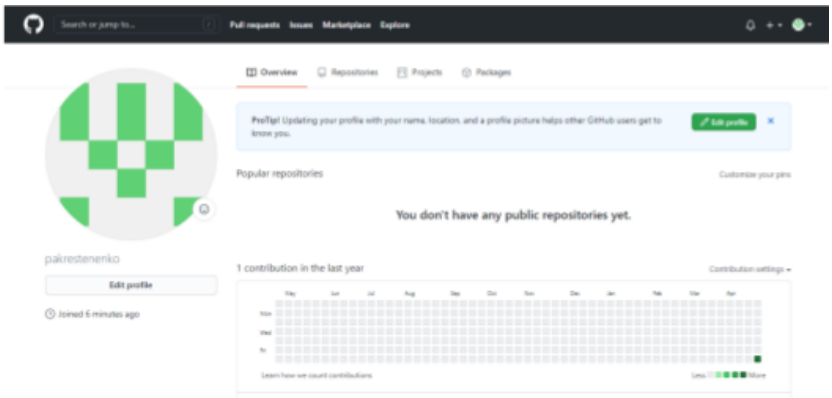

Рис. 3.1: создание учетной записи

• Настраиваю систему контроля версий git. Синхранизирую учётную запись github с компьютером: git config –global user.name “Имя Фамилия” git config –global user.email “work@mail” (рис. -fig. 3.2)

```
[pakrestenenko@pakrestenenko ~]$ git config --global user.name "pakrestenenko"
[pakrestenenko@pakrestenenko ~]$ git config --global user.email "1032201711@pfur.ru"
```

Рис. 3.2: настройка системы контроля

• Создаю новый ключ на github (команда ssh-keygen -C “pakrestenenko 1032201711@pfur.ru”) и привязываю его к копьютеру через консоль.(рис. -fig. 3.3)

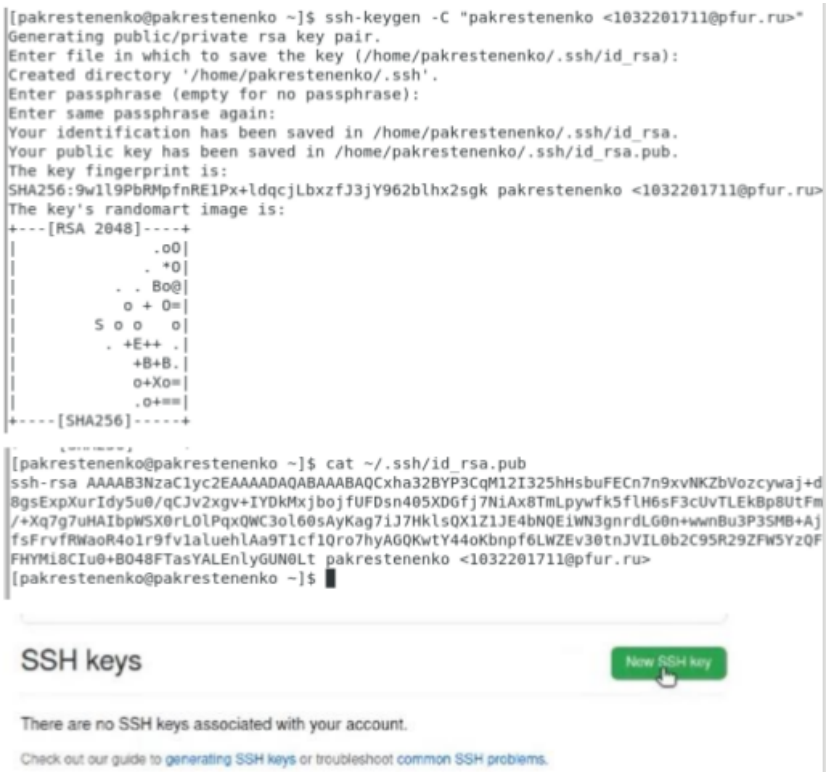

Рис. 3.3: создание нового ключа

(рис. -fig. 3.4)

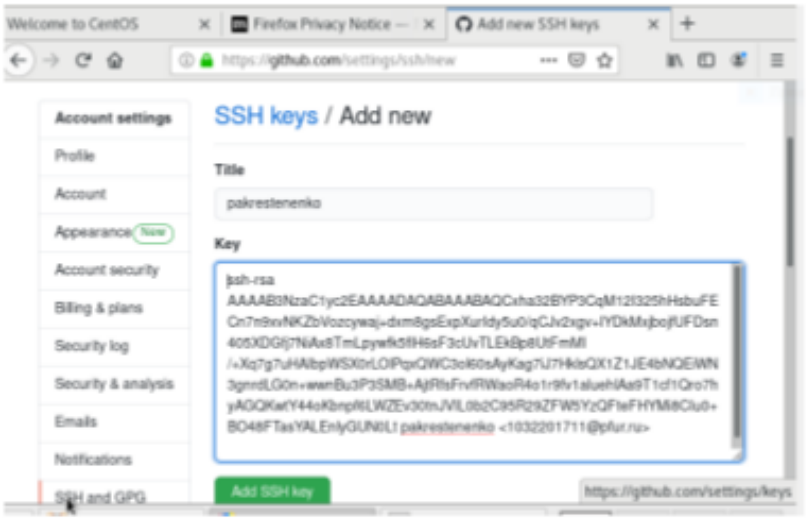

Рис. 3.4: создание нового ключа и привязка

(рис. -fig. 3.5)

Рис. 3.5: создание нового ключа и привязка

1) Подключение репозитория к github • В github захожу в «repositories» и создаю новый репозиторий (имя «laborat2», заголовок для файла README). Копируем в консоль ссылку на репозиторий.(рис. -fig. 3.6)

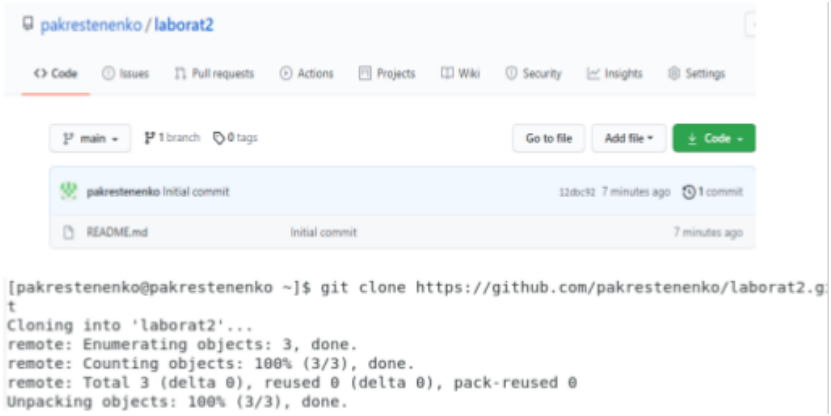

Рис. 3.6: подключение репозитория

• Работаю с каталогом и папками через консоль. Перед тем, как создавать файлы, захожу в репозиорий:(рис. -fig. 3.7)

```
[pakrestenenko@pakrestenenko ~]$ cd laborat2
[pakrestenenko@pakrestenenko laborat2]$ ls
README.md
```

Рис. 3.7: работаю с каталогом

Создаю файлы:(рис. -fig. 3.8)

```
[pakrestenenko@pakrestenenko laborat2]$ mkdir 2020-2021
[pakrestenenko@pakrestenenko laborat2]$ cd 2020-2021

[pakrestenenko@pakrestenenko 2020-2021]$ mkdir may
[pakrestenenko@pakrestenenko 2020-2021]$ cd may
[pakrestenenko@pakrestenenko may]$ mkdir lab02
[pakrestenenko@pakrestenenko may]$ cd lab02
[pakrestenenko@pakrestenenko lab02]$ touch b.txt
```

Рис. 3.8: создание файлов

• Добавляю первый коммит и выкладываю на githup. Для того, чтобыправильно разместить первый коммит, необходимо добавить команду git add . , далее с помощью команды git commit -m “first commit” выкладываем коммит:(рис. -fig. 3.9)

```
[pakrestenenko@pakrestenenko lab02]$ git add .
[pakrestenenko@pakrestenenko lab02]$ git commit -m "first commit"
[main 13c524c] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2020-2021/may/lab02/b.txt
```

Рис. 3.9: добавляем комминт

• Сохраняю первый коммит (git push):(рис. -fig. 3.10)

```
[pakrestenenko@pakrestenenko lab02]$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': pakrestenenko
Password for 'https://pakrestenenko@github.com':
Counting objects: 7, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 390 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/pakrestenenko/laborat2.git
   12dbc92..13c524c  main -> main
```

Рис. 3.10: сохранение первого коммита

2) Первичная конфигурация • Добавляю файл лицензии:(рис. -fig. 3.11)

```
[pakrestenenko@pakrestenenko lab02]$ wget https://creativecommons.org/licenses/by/4.0
egalcode.txt -O LICENSE
--2021-05-01 19:20:04--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.150.1
 104.20.151.16, ...
Подключение к creativecommons.org (creativecommons.org)|172.67.34.140|:443... соедине
е установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

    [ <=>                                   ] 18 657      --.-K/s   за 0,02s

2021-05-01 19:20:06 (1,14 MB/s) - «LICENSE» сохранён [18657]
```

Рис. 3.11: добавление файла лицензии

• Добавляю шаблон игнорируемых файлов. Получаю список имеющихся шаблонов (на скрине представлены не все шаблоны)(рис. -fig. 3.12)

```
[pakrestenenko@pakrestenenko lab02]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratortitanium,appcode,appcode+all,appcode+iml
appengine,aptanastudio,arcanist,archive,archives
archlinuxpackages,aspnetcore,assembler,ate,atmelstudio
ats,audio,automationstudio,autotools,autotools+strict
awr,azurefunctions,backup,ballerina,basercms
basic,batch,bazaar,bazel,bitrise
bitrix,bittorrent,blackbox,bloop,bluej
bookdown,bower,bricxcc,buck,c
c++,cake,cakephp,cakephp2,cakephp3
```

Рис. 3.12: добавление шаблона игнорируемых файлов

• Скачиваю шаблон, например, для C. Также добавляю новые файлы и выполняю коммит:(рис. -fig. 3.13)

```
[pakrestenenko@pakrestenenko lab02]$ git commit -am "Create a template for
[main 053e1a2] Create a template for C
 2 files changed, 455 insertions(+)
 create mode 100644 2020-2021/may/lab02/.gitignore
 create mode 100644 2020-2021/may/lab02/LICENSE
```

Рис. 3.13: скачивание шаблона для C

• Отправляю на github (git push):(рис. -fig. 3.14)

```
[pakrestenenko@pakrestenenko lab02]$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': pakrestenenko
Password for 'https://pakrestenenko@github.com':
Counting objects: 11, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 6.59 KiB | 0 bytes/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/pakrestenenko/laborat2.git
   13c524c..053e1a2  main -> main
```

Рис. 3.14: отправляю на гитхаб

3) Конфигурация git-flow • Инициализирую git-flow, используя команду git flow init -f (префикс для ярлыков установлен в v):(рис. -fig. 3.15)

```
[pakrestenenko@pakrestenenko lab02]$ git flow init -f

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/] v
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
```

Рис. 3.15: конфигурация

• Проверяю, что нахожусь на ветке develop (git branch):(рис. -fig. 3.16)

```
[pakrestenenko@pakrestenenko lab02]$ git branch
* develop
  main
```

Рис. 3.16: проверка

• Создаю релиз с версией 1.0.0:(рис. -fig. 3.17)

```
[pakrestenenko@pakrestenenko lab02]$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish '1.0.0'
```

Рис. 3.17: создание релиза

• 4. Записываю версию и добавляю в индекс: echo“1.0.0” » VERSION git add . git commit -am ‘chore(main): add version’(рис. -fig. 3.18)

```
[pakrestenenko@pakrestenenko lab02]$ echo "1.0.0" >> VERSION
[pakrestenenko@pakrestenenko lab02]$ git add .
[pakrestenenko@pakrestenenko lab02]$ git commit -am 'chore(main): add version'
[release/1.0.0 d43ad53] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 2020-2021/may/lab02/VERSION
```

Рис. 3.18: записываю видео

• Заливаю релизную ветку в основную ветку (команда git flow release finish1.0.0):(рис. -fig. 3.19)

```
[pakrestenenko@pakrestenenko lab02]$ git flow release finish 1.0.0
Switched to branch 'main'
Merge made by the 'recursive' strategy.
 2020-2021/may/lab02/VERSION | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 2020-2021/may/lab02/VERSION
fatal: no tag message?
Tagging failed. Please run finish again to retry.
```

Рис. 3.19: заливаю релизную ветку

• Отправляю данные на github: git push - -all git push - -tags(рис. -fig. 3.20)

```
[pakrestenenko@pakrestenenko lab02]$ git push --all
Username for 'https://github.com': pakrestenenko
Password for 'https://pakrestenenko@github.com':
Counting objects: 11, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 618 bytes | 0 bytes/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/pakrestenenko/laborat2.git
   053e1a2..452da02  main -> main
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0
[pakrestenenko@pakrestenenko lab02]$ git push --tags
Username for 'https://github.com': pakrestenenko
Password for 'https://pakrestenenko@github.com':
Everything up-to-date
```

Рис. 3.20: отправка данных на гитхаб

• Создаю релиз на github. Заходим в «Releases», нажимаю «Создать новый релиз». Захожу в теги и заполняю все поля (теги для версии 1.0.0). После создания тега, автоматически сформируется релиз.(рис. -fig. 3.21)

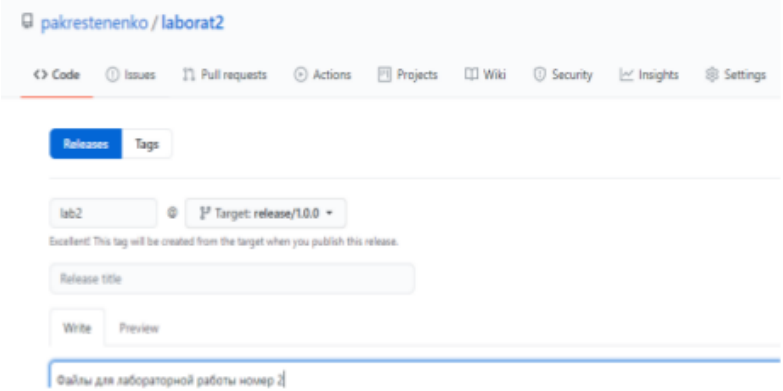

Рис. 3.21: заливаю релизную ветку

Контрольные вопросы: 1.Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с 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 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: git config –global 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 репозиториев

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: о создание основного дерева репозитория: git init о получение обновлений (изменений)текущего дерева из центрального репозитория: git pull о отправка всех произведённых изменений локального дерева в центральный репозиторий: git push о просмотр списка изменённых файлов в текущей директории: git status о просмотр текущих изменения: git diff о добавить все изменённые и/или созданные файлы и/или каталоги: git add . о добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add «имена_файлов» о удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов о сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’о сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit о создание новой ветки, базирующейся натекущей: git checkout -b имя_ветки о 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) о отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки о слияние ветки стекущим деревом: git merge –no-ff имя_ветки о удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки о принудительное удаление локальной ветки: git branch -D имя_ветки о удаление ветки с центрального репозитория: git push origin :имя_ветки 8. Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): git add hello.txt git commit -am ‘Новый файл’ 9. 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 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. Для этого сначала нужно получить список 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

4 Выводы

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий