

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Егорова Юлия Владимировна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	12

Список иллюстраций

3.1	Задание имени и email.	6
3.2	Настройка utf-8.	6
3.3	Задание имени начальной ветки.	6
3.4	Параметр autocrlf.	6
3.5	Параметр safecrlf.	6
3.6	Создание ssh ключей по алгоритму rsa.	7
3.7	Создание ssh ключей по алгоритму ed25519.	7
3.8	Создание ssh ключей по алгоритму ed25519.	7
3.9	Создание pgr ключей.	8
3.10	Создание pgr ключей.	9
3.11	Вывод списка ключей.	9
3.12	Вывод списка ключей.	9
3.13	Настройка автоматических подписей коммитов Git.	9
3.14	Создание каталога.	10
3.15	Добавление ссылки в команду git clone –recursive.	10
3.16	Переход в каталог и удаление из него лишних файлов.	10
3.17	Создание необходимых каталогов.	10
3.18	Отправка файлов на сервер.	11
3.19	Отправка файлов на сервер.	11

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 наGithub.

Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

1. Зададим имя и email владельца репозитория: (рис. 3.1)

```
yvegorova@dk8n59 ~ $ git config --global user.name "Yulia Egorova"  
yvegorova@dk8n59 ~ $ git config --global user.email "egorova21102003@yandex.ru"
```

Рис. 3.1: Задание имени и email.

2. Настроим utf-8 в выводе сообщений git: (рис. 3.2)

```
yvegorova@dk8n59 ~ $ git config --global core.quotePath false
```

Рис. 3.2: Настройка utf-8.

3. Настроим верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master): (рис. 3.3)

```
yvegorova@dk8n59 ~ $ git config --global init.defaultBranch master
```

Рис. 3.3: Задание имени начальной ветки.

4. Введем параметр autocrlf: (рис. 3.4)

```
yvegorova@dk8n59 ~ $ git config --global core.autocrlf input
```

Рис. 3.4: Параметр autocrlf.

5. Введем параметр safecrlf: (рис. 3.5)

```
yvegorova@dk8n59 ~ $ git config --global core.safecrlf warn
```

Рис. 3.5: Параметр safecrlf.

6. Создаем ключи ssh.

По алгоритму rsa с ключом размером 4096 бит: (рис. 3.6)

```
yvegorova@dk8n59 ~ $ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Zby/gJSz01HbZ34P7pbQ1rUmJwxah0mKNvx8EYof134 yvegorova@dk8n59
The key's randomart image is:
+---[RSA 4096]-----+
|
| . .
| . . * +
| = = 0 + .
| . S * B...=
| o @ * .oB.
| * * +o*.o
| o . +oEo
| .o+ .
+---[SHA256]-----+
```

Рис. 3.6: Создание ssh ключей по алгоритму rsa.

По алгоритму ed25519: (рис. 3.7)

```
yvegorova@dk8n59 ~ $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519):
/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519.pub
```

Рис. 3.7: Создание ssh ключей по алгоритму ed25519.

```
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:1A0usVRL47uPmGQW4/CJFFdZn7JjUkIxxQCKY7s yvegorova@dk8n59
The key's randomart image is:
+ [Ed25519 256] +
|
| =U+U+o+
| + -XB+.-
| ++aXa .
| . 3.D .
| S o E.+
| o b o
| .
+---[SHA256]-----+
```

Рис. 3.8: Создание ssh ключей по алгоритму ed25519.

7. Создаем ключи pgr.

Генерируем ключ.

Из предложенных опций выбираем:

тип RSA and RSA;

размер 4096;

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 убедились, что он соответствует адресу, используемому на GitHub.

Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
yvegorova@dk8n59 ~ $ 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) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Egorova Yulia
Адрес электронной почты: egorova21102003@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Egorova Yulia <egorova21102003@yandex.ru>"

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

Рис. 3.9: Создание gpg ключей.


```

Сменить (N)имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, движение мыши, обращение к диску); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, движение мыши, обращение к диску); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
dpg: ключ 5A5BA6BEAC72688B помечен как абсолютно доверенный
dpg: сертификат отложен записи в "/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.gnupg/openpgp-revocs.d/92B340EFB24668B035FC10765A5BA6BEAC72688B.rev".
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-27 [SC]
      92B340EFB24668B035FC10765A5BA6BEAC72688B
uid    Egorova Yulia <egorova21102003@yandex.ru>
sub    rsa4096 2022-04-27 [E]

```

Рис. 3.10: Создание pgp ключей.

8. Выводим список ключей и копируем отпечаток приватного ключа:

```

yvegorova@dkn59 ~ $ 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: 3 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 3u
/afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/.gnupg/pubring.kbx
-----
sec   rsa4096/61367AA9FB665DA4 2022-04-22 [SC]
      C82CBF73B452CAF9C5EF4B5761367AA9FB665DA4
uid    [ абсолютно ] Yulia Egorova <egorova21102003@yandex.ru>
ssb    rsa4096/00B73315878249A5 2022-04-22 [E]

sec   rsa4096/A587EA79694740A4 2022-04-22 [SC]
      A6F18F1F0D496DF068C4A62CA587EA79694740A4
uid    [ абсолютно ] Yulia Egorova <egorova21102003@yandex.ru>
ssb    rsa4096/BAB7596B397E4491 2022-04-22 [E]

sec   rsa4096/5A5BA6BEAC72688B 2022-04-27 [SC]
      92B340EFB24668B035FC10765A5BA6BEAC72688B
uid    [ абсолютно ] Egorova Yulia <egorova21102003@yandex.ru>
ssb    rsa4096/B6636D20323DF858 2022-04-27 [E]

```

Рис. 3.11: Вывод списка ключей.

9. Копируем сгенерированный PGP ключ в буфер обмена:

```

yvegorova@dkn38 ~ $ gpg --armor --export 61367AA9FB665DA4 | xclip -sel clip

```

Рис. 3.12: Вывод списка ключей.

И переходим в настройки GitHub по ссылке, нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода.

10. Используя введенный email, указываем Git и применить его при подписи коммитов:

```

yvegorova@dkn59 ~ $ git config --global user.signingkey 61367AA9FB665DA4
yvegorova@dkn59 ~ $ git config --global commit.gpgsign true
yvegorova@dkn59 ~ $ git config --global gpg.program $(which gpg2)

```

Рис. 3.13: Настройка автоматических подписей коммитов Git.

11. Далее я создала шаблон рабочего пространства по примеру <https://github.com/yamadharma/directory-student-template>.

Создала каталог “Операционные системы” и перешла в него. Затем на Github скопировала ссылку на свой репозиторий по шаблону и добавила ссылку в команду `git clone –recursive`.

```
yvegorova@dk8n59 ~ $ mkdir -p ~/work/study/2021-2022/"Операционные системы"
yvegorova@dk8n59 ~ $ cd ~/work/study/2021-2022/"Операционные системы"
```

Рис. 3.14: Создание каталога.

```
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы $ git clone --recursive git@github.com:yvegorova/study_2021-2022-os-intro.git os-intro
```

Рис. 3.15: Добавление ссылки в команду `git clone –recursive`.

12. После этого я перешла в свой каталог и удалила лишние файлы.

```
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы $ cd os-intro
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ rm package.json
```

Рис. 3.16: Переход в каталог и удаление из него лишних файлов.

13. Создала необходимые каталоги:

```
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ ls
config labs LICENSE Makefile project-personal README.en.md README.git-flow.md README.md structure template
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ make COURSE=os-intro
```

Рис. 3.17: Создание необходимых каталогов.

14. И в конце отправила файлы на сервер:

```

yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ git add .
yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'
[master d70a719] feat(main): make course structure
154 files changed, 17100 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/report.md
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/presentation/Makefile
create mode 100644 labs/lab04/presentation/presentation.md
create mode 100644 labs/lab04/report/Makefile
create mode 100644 labs/lab04/report/bib/cite.bib
create mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab04/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab04/report/report.md
create mode 100644 labs/lab05/presentation/Makefile
create mode 100644 labs/lab05/presentation/presentation.md
create mode 100644 labs/lab05/report/Makefile
create mode 100644 labs/lab05/report/bib/cite.bib
create mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg

```

Рис. 3.18: Отправка файлов на сервер.

```

yvegorova@dk8n59 ~/work/study/2021-2022/Операционные системы/os-intro $ git push
Перечисление объектов: 24, готово.
Подсчет объектов: 100% (24/24), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (20/20), готово.
Запись объектов: 100% (23/23), 266.81 КиБ | 2.04 МиБ/с, готово.
Всего 23 (изменений 5), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:yvegorova/study_2021-2022_os-intro.git
  5346338..d70a719  master -> master

```

Рис. 3.19: Отправка файлов на сервер.

4 Выводы

В ходе данной лабораторной работы я изучила идеологию и применение средств контроля версий. А также освоила умения по работе с git.

Контрольные вопросы:

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Version Control System — это программное обеспечение для облегчения работы с изменяющейся информацией. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Основное дерево проекта обычно 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.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. 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: они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Более 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. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

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. Опишите действия сVCS при единоличной работе с хранилищем

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 настроим 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. Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия work@mail"
```

Ключи сохраняются 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. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечивать удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git.

Основные команды git:

1. **git init** – создание основного дерева репозитория.
2. **git pull** - получение обновлений текущего дерева из центрального репозитория.

3. **git push** – отправка всех произведённых изменений локального дерева в центральный репозиторий.
4. **git status** – просмотр списка изменённых файлов в текущей директории.
5. **git diff** - просмотр текущих изменений.
6. **git add** - добавить все изменённые и/или созданные файлы и/или каталоги.
7. **git add имена_файлов** - добавить конкретные изменённые и/или созданные файлы и/или каталоги.
8. **git rm имена_файлов** - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории).
9. **git commit -am 'Описание коммита'** - сохранить все добавленные изменения и все изменённые файлы.
10. **git commit** - сохранить добавленные изменения с внесением комментария через встроенный редактор.
11. **git checkout -b имя_ветки** –создание новой ветки, базирующейся на текущей.
12. **git checkout имя_ветки** - переключение на некоторую ветку.
13. **git push origin имя_ветки** - отправка изменений конкретной ветки в центральный репозиторий.
14. **git merge --no-ff имя_ветки**- слияние ветки стекущим деревом.
15. **git branch -d имя_ветки** - удаление локальной уже слитой с основным деревом ветки.
16. **git branch -D имя_ветки** - принудительное удаление локальной ветки.
17. **git push origin :имя_ветки** - удаление ветки с центрального репозитория.

18. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

```
git add hello.txt
```

```
git commit -am 'Новый файл.'
```

9. Что такое и зачем могут быть нужны ветви (branches)?

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. 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. Для этого сначала нужно получить списки меняющихся 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
```