Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Лабораторная работа 2

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПМбв-01-19

Студенческий билет №: 1032187017

Кушнирчук Дарья Вадимовна

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

Цель работы

Мы изучим идеологию и применение средств контроля версий, также мы освоим умения по работе с git.

Начало работы

Настроим github.

Для этого предварительно создадим учетную страницу на сайте github.com.

Мы создали УЗ: https://github.com/dariakus; и заполнили необходимые данные. Установим программное обеспечение.

Чтобы установить git из стандартного репозитория CentOS, используем менеджер пакетов yum.

```
--------------------------------------------------------------------------------
Общий размер                                              472 kB/s | 4.5 MB  00:09
Получение ключа из file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Импорт GPG ключа 0xF4A80EB5:
 Владелец   : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
 Отпечаток  : 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
 Пакет      : centos-release-7-9.2009.0.el7.centos.x86_64 (@anaconda)
 Источник   : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Продолжить? [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Обновление  : perl-Git-1.8.3.1-25.el7_9.noarch                                1/4
  Обновление  : git-1.8.3.1-25.el7_9.x86_64                                     2/4
  Очистка     : perl-Git-1.8.3.1-23.el7_8.noarch                                3/4
  Очистка     : git-1.8.3.1-23.el7_8.x86_64                                     4/4
  Проверка    : git-1.8.3.1-25.el7_9.x86_64                                     1/4
  Проверка    : perl-Git-1.8.3.1-25.el7_9.noarch                                2/4
  Проверка    : git-1.8.3.1-23.el7_8.x86_64                                     3/4
  Проверка    : perl-Git-1.8.3.1-23.el7_8.noarch                                4/4

Обновлено:
  git.x86_64 0:1.8.3.1-25.el7_9

Обновлены зависимости:
  perl-Git.noarch 0:1.8.3.1-25.el7_9

Выполнено!
[dvkushnirchuk@dkushnirchuk ~]$
```

Рисунок 1

Базовые настройки git.

Зададим имя и email владельца репозитория

```
[root@dkushnirchuk ~]# git config --global user.name "dkushnirchuk"
[root@dkushnirchuk ~]# git config --global user.email "ksh.d.va@gmail.com"
```

Рисунок 2

Настроим utf-8 в выводе сообщения git

```
[root@dkushnirchuk ~]# git config --global core.quotepath false
```

Рисунок 3

Настроим верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть ее master).

Параметр autocrlf.

Параметр safecrlf.

```
[root@dkushnirchuk ~]# git config --global init.defaultBranch master
[root@dkushnirchuk ~]# git config --global core.autocrlf input
[root@dkushnirchuk ~]# git confug --global core.safecrlf warn
```

Создадим ключи SSH.

По алгоритму rsa с ключем размеров 4096 бит.

По алгоритму ed25519.

```
[root@dkushnirchuk ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:zPtsNN/4L2u0khwDuA6HP7bGnsDotNeXjS1JohknB/I root@dkushnirchuk
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|         .       |
|    . .o. .      |
|     o oS. .     |
|     oE *.+ o .  |
|    o o#.+ X B . |
|   o .+.O+B O =  |
|    o. ++=o. +o+.|
+----[SHA256]-----+
```

```
[root@dkushnirchuk ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519.
Your public key has been saved in /root/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:ow0XcaOcBXBteZc3OgpTqeJ0adqwtTOtCHEWdc07ouI root@dkushnirchuk
The key's randomart image is:
+--[ED25519 256]--+
|      ..=++o+  . |
|       + *=+.oo..|
|        *.+. .o..|
|     . * X . =   |
|      B S * o o  |
|     . X B o     |
|      + + +      |
|       E .       |
|                 |
+----[SHA256]-----+
```

Рисунок 5

Создадим ключи GPG.

Генерируем ключ.

```
[root@dkushnirchuk ~]# gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 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 требуемый тип ключа:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (только для подписи)
   (4) RSA (только для подписи)
Ваш выбор (?-подробнее)? 1
ключи RSA могут иметь длину от 1024 до 4096 бит.
Какой размер ключа необходим? (2048) 4096
Запрашиваемый размер ключа 4096 бит
Выберите срок действия ключа.
         0 = без ограничения срока действительности
      <n>  = срок действительности n дней
      <n>w = срок действительности n недель
      <n>m = срок действительности n месяцев
      <n>y = срок действительности n лет
Ключ действителен до? (0) 0
Ключ не имеет ограничения срока действительности
Все верно? (y/N) y

GnuPG необходимо составить UserID в качестве идентификатора ключа.

Ваше настоящее имя: dkushnirchuk
Email-адрес: ksh.d.va@gmail.com
Комментарий:
Вы выбрали следующий User ID:
    "dkushnirchuk <ksh.d.va@gmail.com>"

Сменить (N)Имя, (C)Комментарий, (E)email-адрес или (O)Принять/(Q)Выход? O
```

– 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. – Комментарий.

Рисунок 7

Добавление GPG **ключа в** GitHub.

Скопируем наш сгенерированный PGP ключ в буфер обмена.

```
[root@dkushnirchuk ~]# gpg --armor --export
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGUPBRcBEAC3X2+jZZGf3Cye04iP7mQGZRwr/sPuSGEDe1NltURTAzLjhXuZ
0dFm7RJ6H+cpNrqtqvAbUJLfWfIPndfwHWLND0rrTs3XpV/15a+fwDdws2Z294U3
6QdQhu2rmEglgBLS2LWCbXRqdGJ3XYGQjqqUJZyHb8QaG6UJ2LIjnRgjSulYI4Mh
5RSfMISzBARR0Qm+xliRVI/kVI3xDfAuQP1QF16wBnzoNHRHhtsDQzGHwqLEZXVl
c4foAFbtf5ttk4CGCEgptFZYcvA7dSAAAZmRWxrWA4Fnx7A7Za9Xyc6g5O8ZAI4W
CQf1drvjqLqwNj6dfclmFaWqtg6JNvRdkFv+kFYrAS9L/W5kw3eyKywlZ+IdqlMI
pVpfetTBrwyacRUvjvKh7KWn10rB1pFRCE+CF89/g2ei6E/6BAsczfRIQv9XfsfD
5znIJJWPOxtRrSbOStxTE8VAnQwh/SDAoSw4KBKLakvJOjDo9BqD1A/TTWXNZ5AO
G+Ta3AAxZ6D8eFQ1K+2MfdH1iFpeu3cjm14gNCOhCHo6PQYaMQmFbgSZXtoE2muo
KGHQl+jWsGfxB7WmLJ6YC67iPZ/LTd/l9cQBeJiUD8ZqGutCTvnopDIAGkTq5QDJ
moIHAANA3OGZGRUsCZLKS0Gzo2GzEY6QqlEp8T2GfEd0aQtgR/OKj0hc8wARAQAB
```

Рисунок 8

Вставим ключ в GitHub.

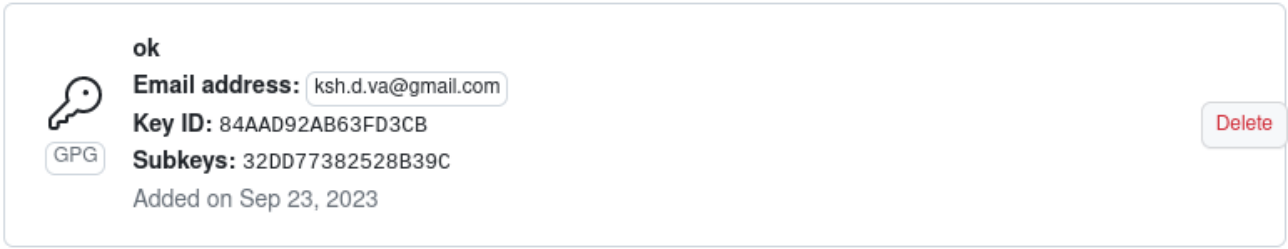

Рисунок 9

Используем введенный email, укажем Git применять его при подписи коммитов.

```
[root@dkushnirchuk ~]# git config --global user.signingkey
[root@dkushnirchuk ~]# git config --global commit.gpgsign true
[root@dkushnirchuk ~]# git config --global gpg.program $(which gpg2)
```

Рисунок 10

Создадим репозиторий курса.

```
[root@dkushnirchuk ~]# mkdir -p ~/work/study/2022-2023/"Операционные системы"
[root@dkushnirchuk ~]# cd ~/work/study/2022-2023/"Операционные системы"
[root@dkushnirchuk Операционные системы]# gh repo create study_2022-2023_os-intro --tem
plate=yamadharma/course-directory-student-template --public
✓ Created repository dariakus/study_2022-2023_os-intro on GitHub
[root@dkushnirchuk Операционные системы]#
[dvkushnirchuk@dkushnirchuk ~]$ git clone --recursive https://github.com/dariaku
s/study_2022-2023_os-intro.git os-intro
Cloning into 'os-intro'...
```

Рисунок 11

Перейдем в каталог курса.

```
[root@dkushnirchuk Операционные системы]# cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

Рисунок 12

Удалим лишние файлы.

```
[root@dkushnirchuk os-intro]# rm package.json
rm: удалить обычный файл «package.json»?
```

Рисунок 13

Создадим необходимые каталоги.

```
[root@dkushnirchuk os-intro]# echo os-intro > COURSE
[root@dkushnirchuk os-intro]# make
```

Рисунок 14

Отправим файлы на сервер.

```
[root@dkushnirchuk os-intro]# git add .
```

```
[root@dkushnirchuk os-intro]# git commit -am 'feat(main): make course structure'
```

```
[root@dkushnirchuk os-intro]# git push
```

Рисунок 15

Вывод

Мы изучили идеологию и применение средств контроля версий. Мы освоили умения по работе с git.

Контрольные вопросы

Контрольные вопросы.

1. *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*

2.
   - *Хранилище (репозиторий) – это система, которая обеспечивает хранение всех существовавших версий файлов.*

- *Commit - запись изменений.*
- *История - список предыдущих изменений.*
- *Рабочая копия – копия файла, с которой непосредственно ведётся работа (находится вне репозитория) С помощью коммитов изменения, внесённые в рабочую копию, заносятся в хранилище. Благодаря истории можно отследить изменения, вносимые в репозиторий. Перед началом работы рабочую копию можно получить из одной из версий, хранящихся в репозитории.*

3. *В централизованных СКВ все файлы хранятся в одном репозитории, и каждый пользователь может вносить изменения. В децентрализованных их несколько, и они могут обмениваться изменениями между собой, а центрального репозитория может не существовать вообще. Среди классических (т.е. централизованных) VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.*
4. *Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).*
5. *Аналогично единоличной работе, но также можно объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику.*
6. *Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Git позволяет создавать локальные репозитории и вносить в них изменения, а также работать с удалёнными репозиториями.*
7.
   - *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 5)просмотр текущих изменения: git diff*
   - *сохранение текущих изменений: а)добавить все изменённые и/или созданные файлы и/или каталоги: git add . б)добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add именафайлов в)удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm именафайлов*

- *сохранение добавленных изменений: а)сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита' б)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 имяветки (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 имяветки*
- *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*

1. *Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am "file.txt was added" Отправим изменения в удалённый репозиторий: git push*
2. *СКВ могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Это удобно при работе над одним проектом нескольких человек, или если вносимые на каждой из ветвей изменения будут разительно отличаться (например, создание программ с разным функционалом на базе одного интерфейса).*
3. *Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять впоследствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы*