

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: *Операционные системы*

Студент: Бондаренко Елизавета

Группа: НПИМбд-01-21

МОСКВА

2022г.

Цель работы:

Целью данной работы является изучение идеологии и применения средств контроля версий, а также освоение умения по работе с git.

Ход работы:

Создаю учётную запись на <https://github.com>. Заполняю основные данные (рис.1).

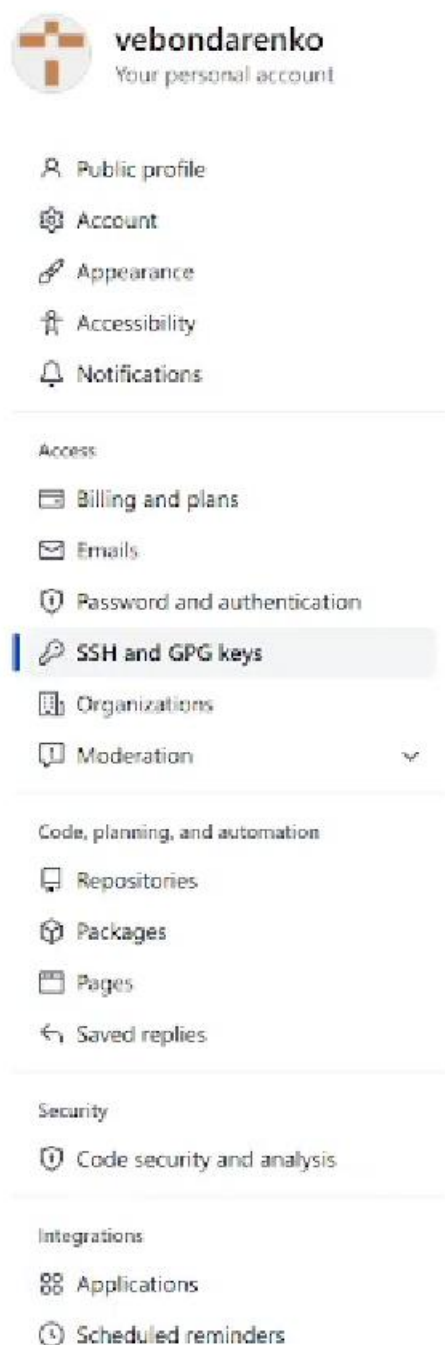


Рис.1

Начинаю базовую настройку git. Задаю имя и email владельца репозитория (рис.2).

```
vebondarenko@dk3n38 ~ $ git config --global user.name "Elizaveta Bondarenko"
vebondarenko@dk3n38 ~ $ git config --global user.email "elisabeta.bondarenko@yandex.ru"
```

Рис.2

Настраиваю utf-8 в выводе сообщений git (рис.3).

```
evbondarenko@dk3n38 ~ $ git config --global core.quotepach false
```

Рис. 3

Настраиваю верификацию и подписание коммитов git. Задаю имя начальной ветки (допустим master) (рис.4). А также параметр autocrlf и параметр safecrlf (рис.5).

```
evbondarenko@dk3n38 ~ $ git config --global init.defaultBranch master
```

Рис.4

```
evbondarenko@dk3n38 ~ $ git config --global core.autocrlf input
evbondarenko@dk3n38 ~ $ git config --global core.safecrlf warn
```

Рис. 5

Создаю ключи ssh – по алгоритму rsa с ключом размером 4096 бит и по алгоритму ed25519 (рис. 6) и (рис.7)

```
evbondarenko@dk3n38 ~ $ 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/e/v/evbondarenko/.ssh/id_rsa):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:7GB46IASCMfuus3zhggVKrbzo9uSpkSS9ziji55URY evbondarenko@dk3n38
The key's randomart image is:
+---[RSA 4096]-----+
| ..O               |
| o=E               |
| o +.              |
| + ..OO .          |
| =+ ++ = $         |
| **= * +           |
| ***= . .          |
| =+o0 =            |
| +**.*..           |
| +-----[SHA256]-----+
```

Рис. 6

```
evbondarenko@dk3n38 ~ $ 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/e/v/evbondarenko/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:KMHcwgoysi7wQy7v3b77qzhuuhiaPm/s+lZVH0dulzk evbondarenko@dk3n38
The key's randomart image is:
+---[ED25519 256]---+
| .O.               |
| + .. +            |
| = * . . . O       |
| o+ . .OO O        |
| o o .OO.SE        |
| o+. OO O          |
| o+=+..            |
| +=.sOO            |
| .+@OOO*=O.        |
| +-----[SHA256]-----+
```

Рис.7

Создаю ключи gpg. Генерирую ключ. Из предложенных опций выбираю: – тип RSA and RSA; – размер 4096; срок действия: значение по умолчанию— 0 (срок

действия не истекает никогда). – GPG запросил личную информацию, которая сохранится в ключе: – Имя. – Адрес электронной почты. – Ввожу email, используемый на GitHub. – Комментарий. Нажимаю клавишу ввода, чтобы оставить это поле пустым (рис. 8).

```
evbondarenko@dk3n38 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.33; 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: Elizaveta
Адрес электронной почты: elisabeta.bondarenko@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Elizaveta <elisabeta.bondarenko@yandex.ru>"

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

Рис. 8

Добавляю GPG ключ в GitHub. Вывожу список ключей и копирую отпечаток приватного ключа. Копирую сгенерированный GPG ключ в буфер обмена (рис. 9). Перехожу в настройки GitHub (<https://github.com/settings/keys>), нажимаю на кнопку New GPG key и вставляю полученный ключ в поле ввода (рис. 10).

```
evbondarenko@dk3n38 ~ $ 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: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/afs/.dk.sci.pfu.edu.ru/home/e/v/evbondarenko/.gnupg/pubring.kbx
-----
sec   rsa4096/EB22C6F02988A8BE 2022-04-21 [SC]
      E92725A0AC242A70410714EEEB22C6F02988A8BE
uid   [ абсолютно ] Elizaveta <elisabeta.bondarenko@yandex.ru>
ssb   rsa4096/19250C49DB65B92F 2022-04-21 [E]


evbondarenko@dk3n38 ~ $ gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
evbondarenko@dk3n38 ~ $ gpg --armor --export <EB22C6F02988A8BE> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
evbondarenko@dk3n38 ~ $ gpg --armor --export EB22C6F02988A8BE | xclip -sel clip
```

Рис. 9

GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



Email address: `elisabeta.bondarenko@yandex.ru`

Key ID: `EB22C6F02988A8BE`

Subkeys: `19250C49DB65B92F`

Added on 21 Apr 2022

[Delete](#)

Рис. 10

Настраиваю автоматические подписи коммитов git. Используя введённый email, указываю Git применять его при подписи коммитов (рис.11).

```
evbondarenko@dk3n38 ~$ git config --global user.signingkey EB22C6F02988A8BE
evbondarenko@dk3n38 ~$ git config --global commit.gpgsign true
evbondarenko@dk3n38 ~$ git config --global gpg.program $(which gpg2)
```

Рис. 11

Используя шаблон для рабочего пространства (<https://github.com/yamadharm/course-directory-student-template>), создаю репозиторий os-intro (рис. 12)

master ▾

1 branch


0 tags

Go to file

Add file ▾

Code ▾

Use this template

 yamadharm chore(submodules): update submodules

274a002 · 4 days ago · 3 commits

config	chore(main): add conventional changelog support	7 days ago
template	chore(submodules): update submodules	4 days ago
.gitattributes	Initial commit	7 days ago
.gitignore	Initial commit	7 days ago
.gitmodules	chore(main): add conventional changelog support	7 days ago
LICENSE	Initial commit	7 days ago
Makefile	chore(main): add conventional changelog support	7 days ago
README.en.md	chore(submodules): update submodules	4 days ago
README.git-flow.md	Initial commit	7 days ago
README.md	chore(main): add conventional changelog support	7 days ago
package.json	chore(main): add conventional changelog support	7 days ago

Рис. 12

Создаю репозиторий курса на основе шаблона. Для этого я создаю каталог «Операционные системы» и перехожу в него . Далее на github я копирую ссылку на свой репозиторий (рис.13) и заполняю репозиторий по шаблону, добавляя ссылку в команду `git clone --recursive`















🔑 master ▾ 🔑 1 branch 🔑 0 tags			Go to file	Add file ▾	Code ▾
 vebondarenko Add files via upload			a152a0e 2 hours ago 4 commits		
	config	Initial commit	2 days ago		
	labs	Add files via upload	2 hours ago		
	project-personal	feat(main): make course structure	2 days ago		
	template	Initial commit	2 days ago		
	.gitattributes	Initial commit	2 days ago		
	.gitignore	Initial commit	2 days ago		
	.gitmodules	Initial commit	2 days ago		
	LICENSE	Initial commit	2 days ago		
	Makefile	Initial commit	2 days ago		
	README.en.md	Initial commit	2 days ago		
	README.git-flow.md	Initial commit	2 days ago		
	README.md	Initial commit	2 days ago		
	structure	feat(main): make course structure	2 days ago		

Рис. 13

Настраиваю каталог курса. Перехожу в каталог курса (рис.14). Удаляю лишние файлы (package.json), но для начала проверяю его наличие в каталоге с помощью команды `ls` (рис.15). Создаю необходимые каталоги (рис.16). Отправляю файлы на сервер

```

evbondarenko@dk3n38 ~ $ cd os-intro
evbondarenko@dk3n38 ~/os-intro $ rm package.json
evbondarenko@dk3n38 ~/os-intro $ ls
config  Makefile  README.git-flow.md  template
LICENSE README.en.md README.md
evbondarenko@dk3n38 ~/os-intro $ make COURSE=os-intro
evbondarenko@dk3n38 ~/os-intro $ ls
config LICENSE project-personal README.git-flow.md structure
labs Makefile README.en.md README.md template

```

Рис.14

```

create mode 100644 labs/lab10/report/report.md
create mode 100644 labs/lab11/presentation/Makefile
create mode 100644 labs/lab11/presentation/presentation.md
create mode 100644 labs/lab11/report/Makefile
create mode 100644 labs/lab11/report/bib/cite.bib
create mode 100644 labs/lab11/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab11/report/pandoc/csl/gost-r-7-0-5-2008-numeric.c
l
create mode 100644 labs/lab11/report/report.md
create mode 100644 labs/lab12/presentation/Makefile
create mode 100644 labs/lab12/presentation/presentation.md
create mode 100644 labs/lab12/report/Makefile
create mode 100644 labs/lab12/report/bib/cite.bib
create mode 100644 labs/lab12/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab12/report/pandoc/csl/gost-r-7-0-5-2008-numeric.c
l
create mode 100644 labs/lab12/report/report.md
create mode 100644 labs/lab13/presentation/Makefile
create mode 100644 labs/lab13/presentation/presentation.md
create mode 100644 labs/lab13/report/Makefile
create mode 100644 labs/lab13/report/bib/cite.bib
create mode 100644 labs/lab13/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab13/report/pandoc/csl/gost-r-7-0-5-2008-numeric.c
l
create mode 100644 labs/lab13/report/report.md
create mode 100644 labs/lab14/presentation/Makefile
create mode 100644 labs/lab14/presentation/presentation.md
create mode 100644 labs/lab14/report/Makefile
create mode 100644 labs/lab14/report/bib/cite.bib
create mode 100644 labs/lab14/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab14/report/pandoc/csl/gost-r-7-0-5-2008-numeric.c
l
create mode 100644 labs/lab14/report/report.md
create mode 100644 labs/lab15/presentation/Makefile
create mode 100644 labs/lab15/presentation/presentation.md
create mode 100644 labs/lab15/report/Makefile
create mode 100644 labs/lab15/report/bib/cite.bib
create mode 100644 labs/lab15/report/image/placeimg_800_600_tech.jpg

```

Рис.15

```

vibondarenko@dk3n38 ~/os-intro $ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.53 КиБ | 2.44 МБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно испо
льзовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:vibondarenko/-os-intro.git
 e528c7d..20a77af master -> master

```

Рис.16

Контрольные вопросы:

1) Version Control System — программное обеспечение для облегчения работы с изменяющейся информацией. 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) 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:
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 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) 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) 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 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 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) Исползования 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 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
```

Вывод: В ходе выполнения данной лабораторной работы я приобрела практические навыки работы с github, научилась создавать репозитории и размещать файлы в них, что упростит работу со следующими лабораторными работами и поможет структурировать информацию, а также я изучила идеологию применения средств контроля версий.