

# Отчёт по лабораторной работе №2

Простейший вариант

Явкина Анастасия Юрьевна

# Список иллюстраций

## Лабораторная работа №3

Цель: Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

Ход работы:

1. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория.

```
ayyavkina@dk8n62 ~ $ git config --global user.email "yavkina.anastasiya@mail.ru"
ayyavkina@dk8n62 ~ $ git config --global user.name "Anastasiya"
ayyavkina@dk8n62 ~ $
```

2. Настраиваем utf-8 в выводе сообщений git. Настраиваем верификацию и подписание коммитов git. Задаем имя начальной ветки (будем называть её master). Параметр autocrlf. Параметр safecrlf.

```
ayyavkina@dk8n62 ~ $ git config --global core.quotepath false
ayyavkina@dk8n62 ~ $ git config --global init.defaultBranch master
ayyavkina@dk8n62 ~ $ git config --global core.autocrlf input
ayyavkina@dk8n62 ~ $ git config --global core.safecrlf warn
```

3. Создаем ключ SSH по алгоритму rsa с ключём размером 4096 бит.

```
ayyavkina@dk8n62 ~ $ 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/a/y/ayyavkina/.ssh/id_rsa):
```

4. Скопировав из локальной консоли ключ в буфер обмена вставляем ключ в появившееся на сайте поле с помощью команды `cat ~/.ssh/id_rsa.pub | xclip -sel clip`.

5. Создаем ключи pgr. Генерируем ключ.

```
ayyavkina@dk8n62 ~ $ 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) Именуемый на карте ключ
Ваш выбор? 1
Введите длину RSA: может быть от 1024 до 4096
```

6. Выводим список ключей и копируем отпечаток приватного ключа.

```
ayyavkina@dk8n62 ~ $ 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/a/y/ayyavkina/.gnupg/pubring.kbx
-----
sec rsa4096/352093396CE0CFCE 2022-04-29 [SC]
    DD79509D7CA1DAE6E355257C352093396CE0CFCE
uid          [ абсолютно ] Anastasiya <yavkina.anastasiya2003@mail.ru>
ssb rsa4096/86AEF1B49973F637 2022-04-29 [E]
```

7. Копируем сгенерированный PGP ключ в буфер обмена. Вставляем ключ. Также вставляем SSH ключ.

```
ayyavkina@dk8n62 ~ $ gpg --armor --export 352093396CE0CFCE | xclip -sel clip
ayyavkina@dk8n62 ~ $ git config --global user.signingkey 352093396CE0CFCE
ayyavkina@dk8n62 ~ $ git config --global commit.gpgsign true
ayyavkina@dk8n62 ~ $ git config --global gpg.program $(which gpg2)
ayyavkina@dk8n62 ~ $ cd /work/study/2021-2022
bash: cd: /work/study/2021-2022: Нет такого файла или каталога
```

8. Используя введенный email, указываем Git применять его при подписи коммитов.

## SSH keys

[New SSH key](#)

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



**ayyavkina@dk8n62**  
SHA256:wGS8jP70AAVWmuYQVC/fk62gszJvae4SkTMBrfefKEw  
Added on 29 Apr 2022  
Never used — Read/write


Delete

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH problems](#).

## 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:** yavkina.anastasiya2003@mail.ru  
**Key ID:** 352893396CE0CFCE  
**Subkeys:** 86AEF1849973F637  
Added on 29 Apr 2022

Delete

[Learn how to generate a GPG key and add it to your account](#)

9. Создаем шаблон рабочего пространства. Для 2021–2022 учебного года и предмета «Операционные системы» (код предмета os-intro) созданием репозиторий. Переходим в каталог курса. Удаляем лишние файлы. Удалите лишние файлы. Отправляем файлы на сервер.

```
ayyavkina@dk8n62 ~ $ git config --global user.email "yavkina.anastasiya@mail.ru"
ayyavkina@dk8n62 ~ $ git config --global user.name "Anastasiya"
ayyavkina@dk8n62 ~ $
ayyavkina@dk8n62 ~ $ git config --global core.quotepath false
ayyavkina@dk8n62 ~ $ git config --global init.defaultBranch master
ayyavkina@dk8n62 ~ $ git config --global core.autocrlf input
ayyavkina@dk8n62 ~ $ git config --global core.safecrlf warn
```

Вывод: Я научилась оформлять файлы в Markdown.

Контрольные вопросы: 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 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 сохранятся 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 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

натекущей: `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 добавлении в репозиторий типов файлов в файл `.gitignore` с помощью `ервисов`. 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`

## Список литературы