Отчет по лабораторной работе №2

дисциплина: "Операционные системы"

Студент: Куприяненко Мария Сергее

###

Группа: НПМбв02-20

Цель работы

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.

Ход работы

1. Установка git

```shell
dnf install git
```

2. Установка gh

```shell
dnf install gh
```

3. Базовая настройка git

- Зададим имя и email владельца репозитория:

```shell
git config --global user.name "Name Surname"
git config --global user.email "work@mail"
```

- Настроим utf-8 в выводе сообщений git:

```shell
git config --global core.quotepath false
```

- Настройте верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG).

```shell
gpg --full-generate-key
```

- Зададим имя начальной ветки (будем называть её master):

```shell
git config --global init.defaultBranch master
```

- Параметр autocrlf:

```shell
git config --global core.autocrlf input
```

- Параметр safecrlf:

```shell
```

```
git config --global core.safecrlf warn
```

4. Создайте ключи ssh

по алгоритму rsa с ключём размером 4096 бит:

```shell
ssh-keygen -t rsa -b 4096
```

по алгоритму ed25519:

```shell
ssh-keygen -t ed25519
```

5. Создайте ключи pgp

```shell
gpg --full-generate-key
```

7. Добавление PGP ключа в GitHub

```shell
gpg --armor --export <PGP Fingerprint> | xclip -sel clip
```

Контрольные вопросы

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

``` shell

программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления

```

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

``` shell

репозиторий (repository) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются

commit - операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию

история - все версии репозитория

рабочая копия - копия проекта, в которую вносятся изменения и производится работа

```

Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

``` shell

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они
```

(изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

```

Опишите действия с VCS при единоличной работе с хранилищем.

Опишите порядок работы с общим хранилищем VCS.

Каковы основные задачи, решаемые инструментальным средством git?

``` shell
Возврат к любой версии кода из прошлого.

Просмотр истории изменений.

Совместная работа без боязни потерять данные или затереть чужую работу.
```

Назовите и дайте краткую характеристику командам git.

Создание основного дерева репозитория:

``` shell
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 с текущим 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

```

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

``` shell

Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку. Она называется master веткой.

```

Как и зачем можно игнорировать некоторые файлы при commit?

``` shell

.gitignore файл в корневом каталоге репозитория, чтобы сообщить Git, какие файлы и каталоги игнорировать при фиксации

```
```