># Российский университит дружбы народов

> #### Факультет физико-математических и естественных наук

> #### Лабораторная работа № 3 . markdown

* > Имя : исса гадир

* > Студенческий билет : 1032218267 * >

* > Группа : нфибд-01-21

Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

 ## **`Домашнее задани`**

 `Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.`

1 : Создать базовую конфигурацию для работы с git :

* Зададим имя и email владельца репозитория:

* Настроим utf-8 в выводе сообщений git:

* Настройте верификацию и подписание коммитов git.

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

* Параметр autocrlf:

* Параметр safecrlf:

<img src="https://i.ibb.co/3CCKP1Y/1-1.png" alt="1-1" border="0">

2 : Создать ключ SSH :

по алгоритму rsa с ключём размером 4096 бит:

по алгоритму ed25519 :

3 : Создать ключ PGP:

* 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.

* При вводе email убедитесь, что он соответствует адресу, используемому на

* GitHub.

* Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы

* оставить это поле пустым.

Добавление PGP ключа в GitHub :

* Выводим список ключей и копируем отпечаток приватного ключа:

* Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

– Формат строки:

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]

ID_ключа

– Скопируйте ваш сгенерированный PGP ключ в буфер обмена:

Перейдите в настройки GitHub (https://github.com/settings/keys), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода.

<img src="https://i.ibb.co/K6F7hWJ/5.png" alt="5" border="0">

4 : Настройка автоматических подписей коммитов git :

Используя введёный email, укажите Git применять его при подписи коммитов:

Настройка gh:

Для начала необходимо авторизоваться

* Утилита задаст несколько наводящих вопросов.

* Авторизоваться можно через броузер

<img src="https://i.ibb.co/K7c9fky/6.png" alt="6" border="0">

5 : Сознание репозитория курса на основе шаблона

Необходимо создать шаблон рабочего пространства.

`Например`-> для 2021–2022 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

<img src="https://i.ibb.co/1fsC9qd/7.png" alt="7" border="0">

* Настройка каталога курса

* Перейдите в каталог курса

* Удалите лишние файлы

* Создайте необходимые каталоги

* Отправьте файлы на сервер

<img src="https://i.ibb.co/8KgmNb7/8-1.png" alt="8-1" border="0">

<img src="https://i.ibb.co/1dfQ5KY/8-2.png" alt="8-2" border="0">

<img src="https://i.ibb.co/Wv0VpCS/8-3.png" alt="8-3" border="0">

<img src="https://i.ibb.co/410fy7x/8-4.png" alt="8-4" border="0">

<img src="https://i.ibb.co/mSVy0Lj/9.png" alt="9" border="0">

Контрольные вопросы

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

Система управления версиями (также используется определение «система контроля версий[1]», от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM).

Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Программное обеспечение Википедии ведёт историю изменений для всех её статей, используя методы, аналогичные тем, которые применяются в системах управления версиями.

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

* Начало работы с проектом:Первым действием, которое должен выполнить разработчик, является извлечение рабочей копии проекта или той его части, с которой предстоит работать. Это действие выполняется с помощью команды извлечения версии (обычно checkout или clone). Разработчик задаёт версию, которая должна быть скопирована, по умолчанию обычно копируется последняя (или выбранная администратором в качестве основной) версия.

* Ежедневный цикл работы:При некоторых вариациях, определяемых особенностями системы и деталями принятого технологического процесса, обычный цикл работы разработчика в течение рабочего дня выглядит следующим образом.

* Обновление рабочей копии:По мере внесения изменений в основную версию проекта рабочая копия на компьютере разработчика стареет: расхождение её с основной версией проекта увеличивается. Это повышает риск возникновения конфликтных изменений

* Модификация проекта:Разработчик модифицирует проект, изменяя входящие в него файлы в рабочей копии в соответствии с проектным заданием

* Фиксация изменений:авершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер (либо в основную ветвь, если работа над заданием полностью завершена, либо в отдельную ветвь разработки данного задания).

`3` : Что представляют собой и чем отличаются централизованные и децентрализованные VCS?

Приведите примеры VCS каждого вида.
Централизованные системы контроля версий
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
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”.

`4`. Опишите действия с VCS при единоличной работе с хранилищем.

создать папку со всеми необходимыми внешними включениями для каждой из двух программ в этом хранилище. В этой папке также были созданы pro-файлы для сборки общего проекта. С учетом выбранной модели, один проект — одно хранилище, никаких негативных последствий от такого решения в дальнейшей разработке мы не испытали.
Если используется подход один проект — одно хранилище, то папки trunk, tags, branches лучше размещать только в корне хранилища.

`5`. Опишите порядок работы с общим хранилищем VCS.

Совместная работа над проектом небольшой территориально распределённой группы разработчиков без выделения общих ресурсов. Как и в предыдущем случае, реализуется схема работы без главного сервера, а актуальность репозиториев поддерживается периодическими синхронизациями по схеме «каждый с каждым».

`6`. Каковы основные задачи, решаемые инструментальным средством git?

В процессе разработки ПО значимая роль отводится сотрудничеству. В большинстве случаев деятельность разработчика включает работу в команде и совместное использование проекта с другими специалистами. Практический опыт использования системы контроля версий не просто важен, а ВАЖЕН для всех, кто намерен заниматься разработкой ПО. В то же время будет сложно привить навык использования контроля версий тем начинающим программистам, которые во время рабочего процесса позволяют коду изливаться из них бурным потоком, вместо того чтобы придержать обороты и размещать его по частям.

`7`. Назовите и дайте краткую характеристику командам git.

Зададим имя и email владельца репозитория:

Настроим utf-8 в выводе сообщений git:

Настройте верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть её master):

* Параметр autocrlf

* Параметр safecrlf

`8`. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Создадим локальный репозиторий.

Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория

и настроив utf-8 в выводе сообщений git

~/tutorial, необходимо ввести в командной строке

После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений.

Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий

Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии

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

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. Она считается центральной веткой, т.е. в ней содержится основной код приложения

`10`. Как и зачем можно игнорировать некоторые файлы при commit?

Можно принудительно сделать коммит игнорируемого файла в репозиторий с помощью команды git add с параметром -f (или --force)

Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Вот некоторые распространенные примеры таких файлов

Выводы:

Я научился создавать отчеты с помощью языка разметки Markdown Light.