# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

#### Факультет физико-математических и естественных наук

### ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

*дисциплина: Операционные системы*

Студент: яссин оулед салем

Группа: НПИбд 02-20

#### Цель работы:

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

#### 3.3. Задание.

– Сделайте отчёт по предыдущей лабораторной работе в формате Markdown. – В качестве отчёта просьба предоставить отчёты в 3 форматах: pdf, docx и md

#### Выполнение задания

#### Оформление лабораторной работы №2 на Markdown.

#### Цель:

Изучить идеологию и применение средств контроля версий. #### Ход работы: 1. Создал учетную запись на github. GITHUB.png

1. Обозначил рабочий каталог как test, создав ее командой mkdir. После перешел в данный каталог командой cd. Инициализировал систему git командой git init. Создаю заготовку для файла README.md: echo "# lab02" >> README.md git add README.md – Делаю первый коммит и выкладываем на github: git commit -m "first commit” git remote add origin git@github.com:/sciproc-intro.git git push -u origin master
2. Добавил файл лицензий:
3. Для начало просмотрим список имеющихся шаблонов
4. Теперь скачиваем шаблон для С
5. Добавил новые файлы и выполнил коммит
6. Отправим на github
7. Инициализировал git-flow, установив префикс для ярлыков в v.
8. Проверяю, что я нахожусь на ветке develop.
9. Создаю релиз с версией 1.0.0
10. Записал версию
11. Добавил в индекс
12. Залил релизную ветку в основную ветку
13. Отправил данные на github
14. Проверил

#### Вывод:

Изучил и понял как работать с системой контроля версий, с помощью командной строки, а именно с Git. Разобрал команды.

Контрольные вопросы:

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

* Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение для облегчения работы с изменяющейся информацией. VCS нужны для хранения полной истории изменений; Описания причин всех производимых изменений; Отката изменений, если что-то пошло не так; Поиска причины и ответственного за появления ошибок в программе; Совместной работы группы над одним проектом; Возможности изменять код, не мешая работе других пользователей.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозитарий, — место хранения файлов и их версий, служебной информации. Версия (revision), или ревизия, — состояние всего хранилища или отдельных файлов в момент времени («пункт истории»). Commit («[трудовой] вклад», не переводится) — процесс создания новой версии; иногда синоним версии. Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.
2. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. 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. Пример: CVS- одна из первых систем второго поколения (1986г.). Обладает множеством недостатков и считается устаревшей.

* Децентрализованные системы контроля версий, в отличие от централизованной модели, может существовать несколько экземпляров репозитория, которые время от времени синхронизируются между собой. Пример: Git- распределенная система управления версиями, созданная Л. Торвальдсом для управления разработкой ядра Linux.

Отличия между централизованными и децентрализованными VCS. Централизованные: • Простота использования. • Вся история — всегда в едином общем хранилище. • Нужно подключение к сети. • Резервное копирование нужно только одному хранилищу. • Удобство разделения прав доступа к хранилищу. • Почти все изменения навсегда попадают в общее хранилище. Децентрализованные: • Двухфазный commit: 1) запись в локальную историю; 2) пересылка изменений другим. • Подключение к сети не нужно. • Локальные хранилища могут служить резервными копиями. • Локальное хранилище контролирует его владелец, • но общее — администратор. • Возможна правка локальной истории перед отправкой на сервер.

1. Опишите действия с VCS при единоличной работе с хранилищем. 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.
2. Опишите порядок работы с общим хранилищем VCS. Работа с общим хранилищем выглядит так:

KONTR.-VOPROS--5.png

KONTR.-VOPROS--5.png

1. Каковы основные задачи, решаемые инструментальным средством git? Задачи решаемые git: Как не потерять файлы с исходным кодом? Как защититься от случайных исправлений и удалений? Как отменить изменения, если они оказались некорректными? Как одновременно поддерживать рабочую версию и разработку новой?
2. Назовите и дайте краткую характеристику командам git.

* add - добавить файл или папку в репозиторий git;
* am - применить все патчи из email;
* archive - создать архив файлов;
* bisect - использовать бинарный поиск для поиска нужного коммита;
* branch - управление ветками проекта;
* bundle - перемещение объектов и ссылок в архиве;
* checkout - переключение между ветками;
* cherry-pick - внести изменения в уже существующие коммиты;
* clean - удалить все неотслеживаемые файлы и папки проекта;
* clone - создать копию удаленного репозитория в папку;
* commit - сохранить изменения в репозиторий;
* diff - посмотреть изменения между коммитами;
* fetch - скачать удаленный репозиторий;
* init - создать репозиторий;
* merge - объединить две ветви;
* pull - интегрировать удаленный репозиторий с локальным;
* push - отправить изменения в удаленный репозиторий;
* tag - управление тегами;
* worktree - управление деревями разработки.

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.
2. 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.
3. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки. Некоторые общие примеры включают в себя: Файлы времени выполнения, такие как журнал, блокировка, кэш или временные файлы. Файлы с конфиденциальной информацией, такой как пароли или ключи API. Скомпилированный код, такой как .class или .o. Каталоги зависимостей, такие как /vendor или /node\_modules. Создавать папки, такие как /public, /out или /dist. Системные файлы, такие как .DS\_Store или Thumbs.db Конфигурационные файлы IDE или текстового редактора.

### Вывод к 3 лабораторной работе.

Я изучив базовые сведения о Markdown, научился оформлять отчеты. Освоил синтаксис данного языка разметки. Выполнил 2 лабораторную работу на Markdown.