Лабораторная работа №2

Контроль версий git

Федюшина Ярослава Андреевна НБИбд-02-21

Содержание

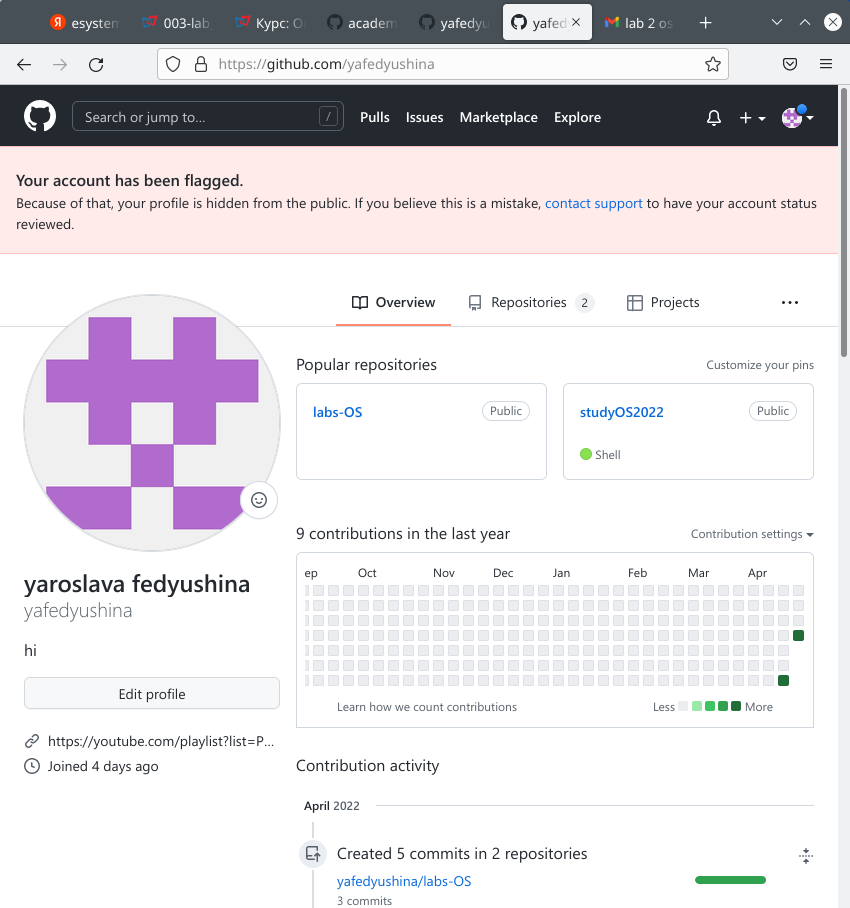
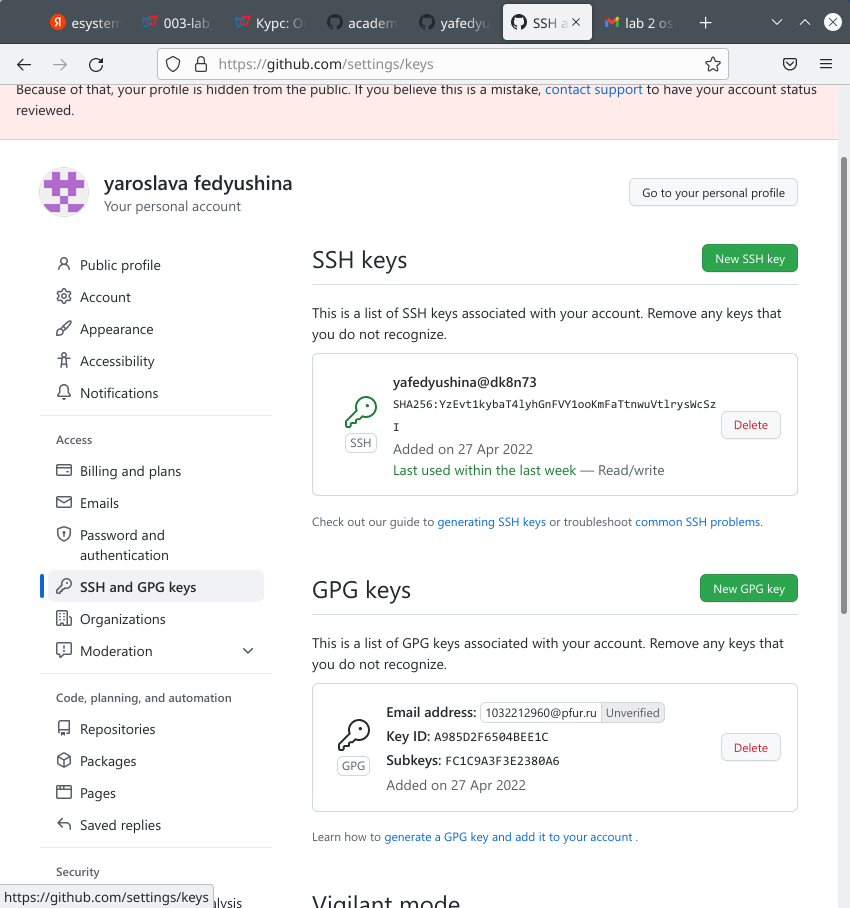
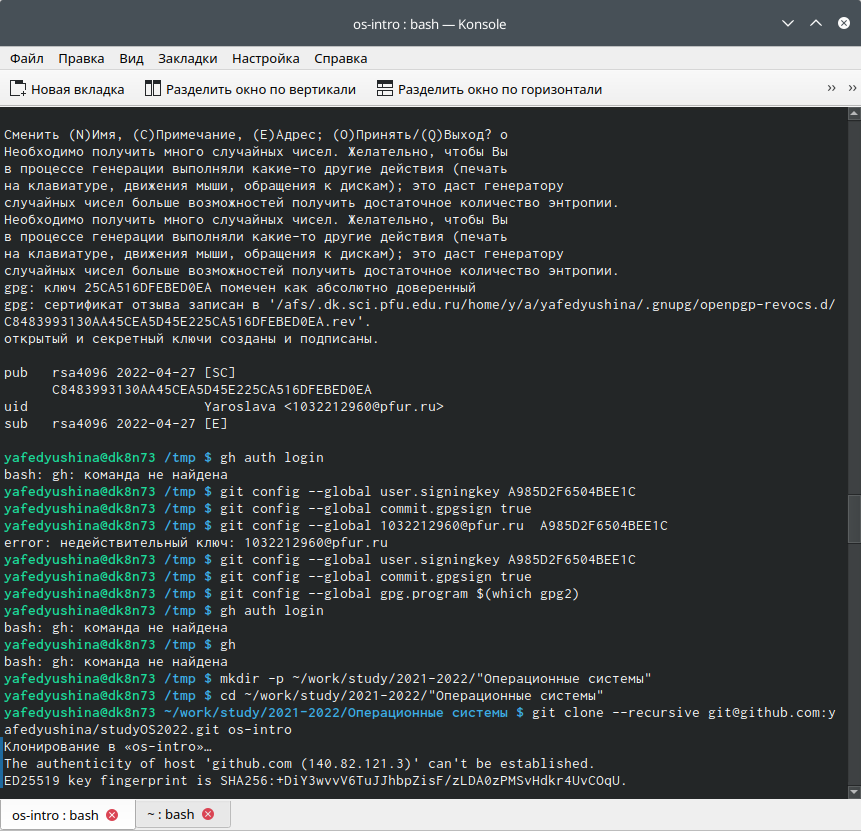
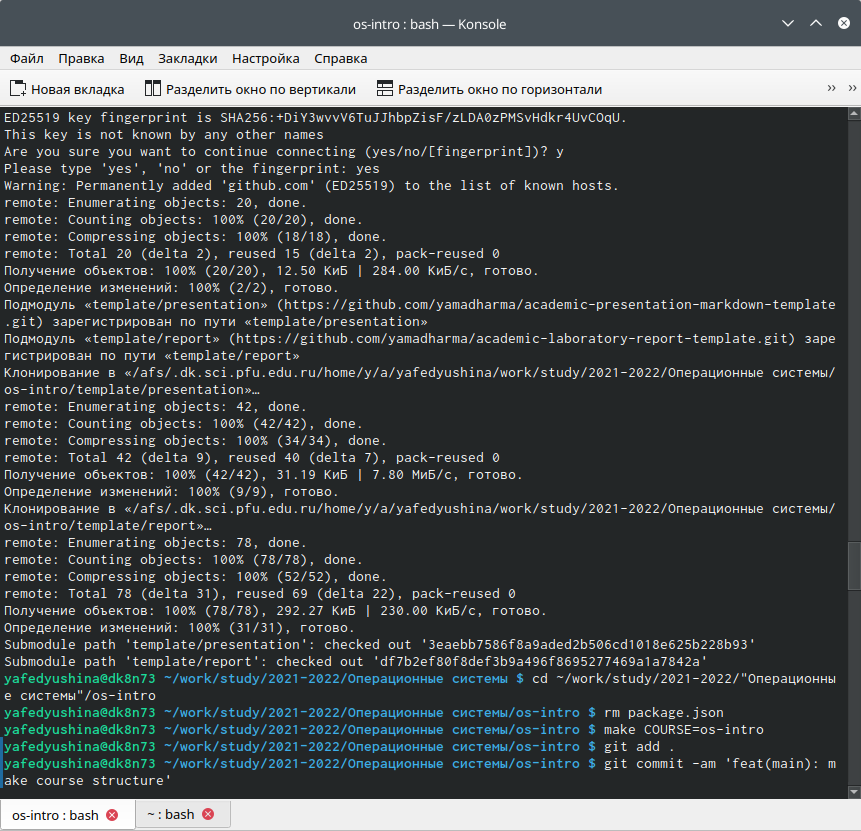
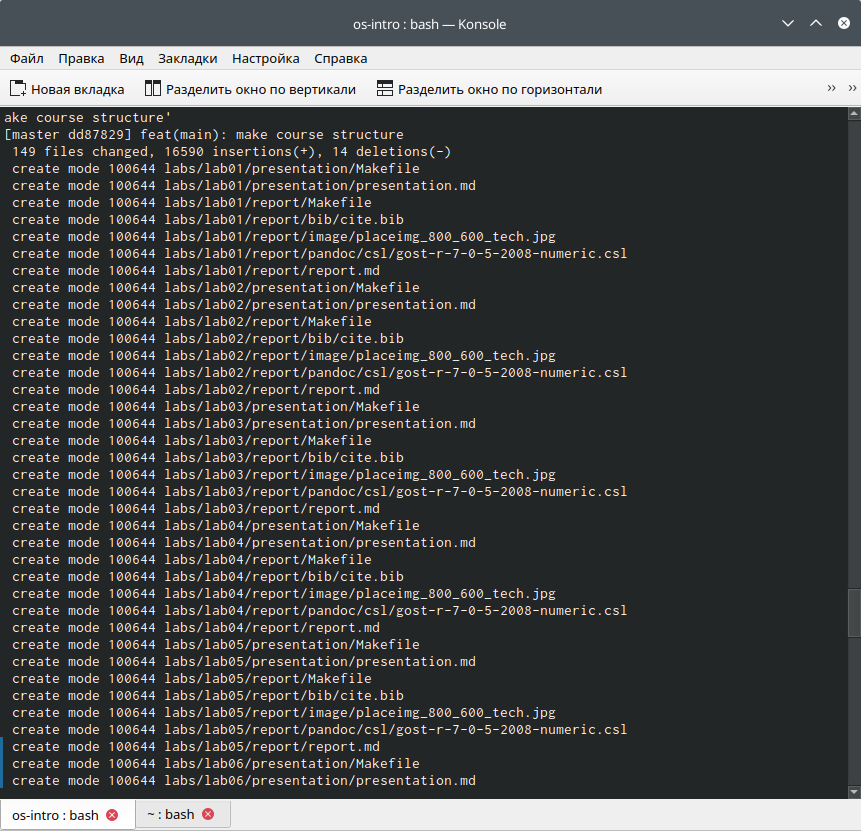
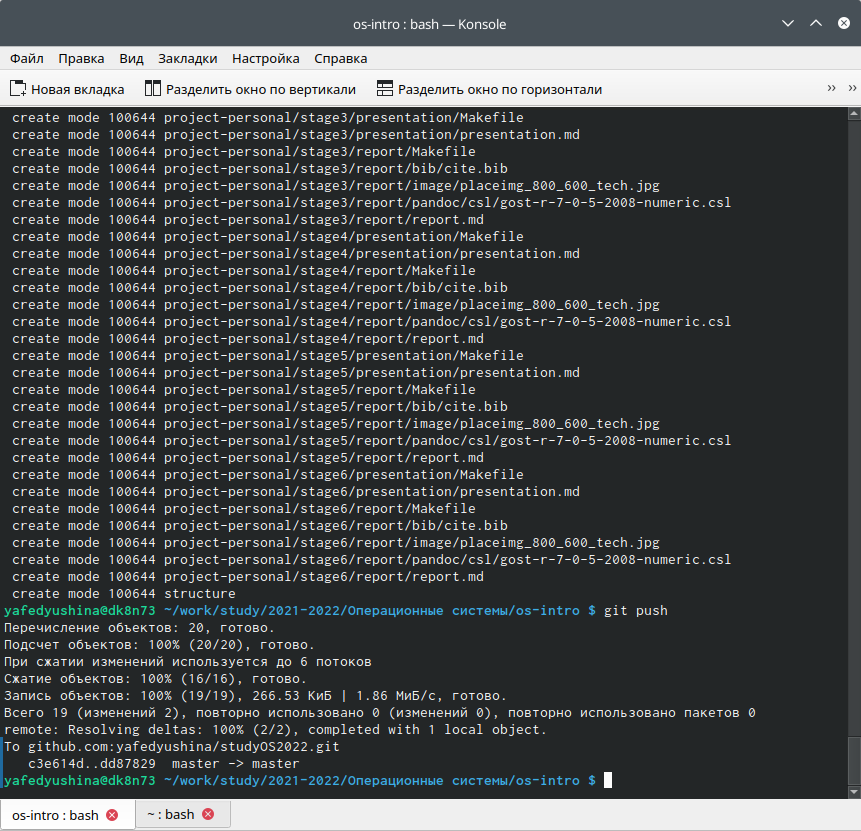
# 1 Цель работы

Целью данной работы является научиться оформлять отчёты с помощью легковесного языка Markdown

# 2 Задание

1. Сделать отчёт по предыдущей лабораторной работе (№2) в формате Markdown
2. В качестве отчёта просьба предоставить отчёты в 3 форматах: pdf, docx и md (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)

# 3 Выполнение лабораторной работы

Создаём аккаунт на Github. У меня уже был создан, делаю скрин уже готового аккаунта  Создаём ключи SSH и GPG   Загружаем шаблон репозитория и синхронизация   Подготовка репозитория и коммит изменений 

# 4 Контрольные вопросы

1. Системы контроля версий (Version Control System, VCS) применяются при работе несколь- ких 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)хранилище - пространство на накопителе где расположен репозиторий commit - сохранение состояния хранилища история - список изменений хранилища рабочая копия - локальная копия сетевого репозитория, в которой работает программист
2. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном эк- земпляре 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 ре- позиторий (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, выби- рается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.
3. Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы. 5)Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя сливается с основной веткой проекта
   1. Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches). 2.Работа с изменениями: выявление (diff), слияние (patch, merge). 3.Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер
4. git config - установка параметров git status - полный список изменений файлов, ожидающих коммита git add . - сделать все измененные файлы готовыми для коммита. git commit -m “[descriptive message]” - записать изменения с заданным сооб- щением. git branch - список всех локальных веток в текущей директории. git checkout [branch-name] - переключиться на указанную ветку и обновить рабочую директорию. git merge [branch] — соединить изменения в текущей ветке с изменениями из заданной. git push - запушить текущую ветку в удаленную ветку. git pull - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.
5. git remote add [имя] [url] — добавляет удалённый репозиторий с заданным именем; git remote remove [имя] — удаляет удалённый репозиторий с заданным именем; git remote rename [старое имя] [новое имя] — переименовывает удалённый репозиторий; git remote set-url [имя] [url] — присваивает репозиторию с именем новый адрес; 19 git remote show [имя] — показывает информацию о репозитории.
6. Ветвление — это возможность работать над разными версиями проекта: вме- сто одного списка с упорядоченными коммитами история будет расходиться в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется master, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью
7. Нет проблем если шаблон для игнорирования подходит для файла под контро- лем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют пока- зываются неизвестные файлы, или просто игнорируются. Файл git.rignore обычно должен быть под контролем версий, что бы новые копии ветки видели такие же шаблоны: git add . gitignore git commit -m “Добавлены шаблоны для игно- рирования”. Многие деревья с исходным кодом содержат файлы, которые не нужно хранить под контролем версий, например, резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать bzr игнорировать их добавив их в файл в корне рабочего дере- ва. Этот файл содержит список шаблонов файлов, по одному в каждой строчке. Обычное содержимое может быть таким: .o ~ .tmp .py [ co ] Если шаблон содер- жит слеш, то он будет сопоставлен с полным путем начиная от корня рабочего дерева; иначе он сопоставляется только с именем файла. Таким образом пример выше игнорирует файлы с расширением .o во всех подкаталогах, но пример ниже игнорирует только config.h в корне рабочего дерева и HTML файлы в каталоге doc/: ./config.h doc/.html Для получения списка файлов которые игнорируются и со- 20 ответствующих им шаблонов используйте команду git ignored : $ git ignored config.h ./config.h configure.in~ ~ $

# 5 Выводы

В ходе выполнения лабораторной работы №3 я научилась оформлять отчёты в формате легковесного языка разметки Markdown

# Список литературы

1. Лекция Системы контроля версий [Электронный ресурс]. URL: http://uii. mpei.ru/study/courses/sdt/16/lecture02.2\_vcs.slides.pdf.