**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

*дисциплина: Операционные системы*

**Студент**: Доленко Дарья

**Группа**: НБИбд-01-21

**Ст.билет №:** 1032211210

Москва

2022 г.

**Цель работы:** Изучение идеологии и применения средств контроля версия; освоение умений по работе с git.

**Последовательность выполнения работы:**

Настраиваю github – создаю учетную запись на <https://github.com>.

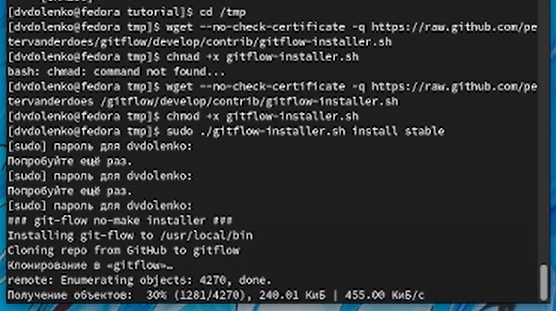
Регистрация на сайте не прогрузилась на компьютере: я произвела её с телефона.

Устанавливаю git-flow в Fedora Linux: это программное обеспечение удалено из репозитория, необходимо устанавливать его вручную с помощью команд:

1. cd /tmp
2. wget --no-check-certificate -q

https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh

1. chmod +x gitflow-installer.sh
2. sudo ./gitflow-installer.sh install stable



Устанавливаю gh в Fedora Linux при помощи команды sudo dnf install gh:



Произвожу базовую настройку git:

* Задаю имя и email владельца репозитория при помощи команд:

git config --global user.name "Name Surname"

git config --global user.email "work@mail"

* Настраиваю utf-8 в выводе сообщений git при помощи команды:

git config --global core.quotepath false

* Настраиваю верификацию и подписание коммитов git, задаю имя начальной ветки master при помощи команды:

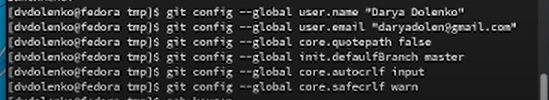
git config --global init.defaultBranch master

* Настраиваю параметр autocrlf при помощи команды:

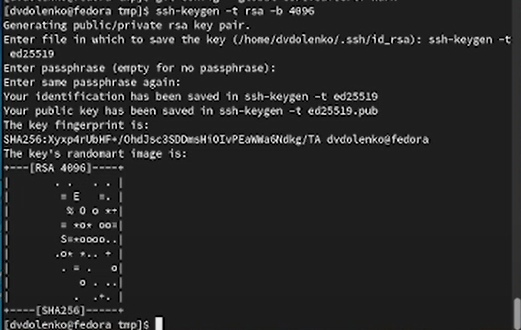
git config --global core.autocrlf input

* Настраиваю параметр safecrlf при помощи команды:

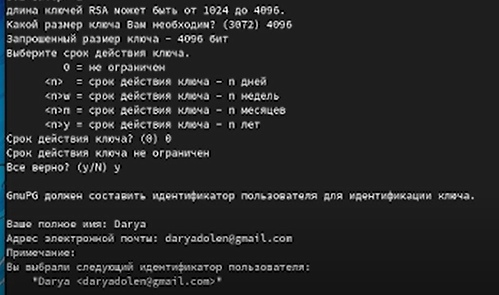
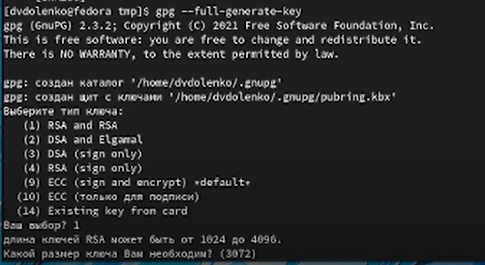
git config --global core.safecrlf warn



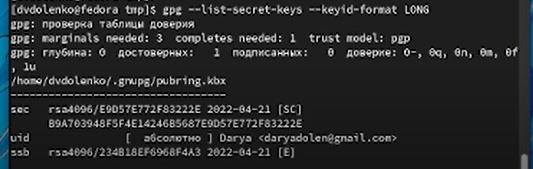
По алгоритму rsa создаю ключи ssh:

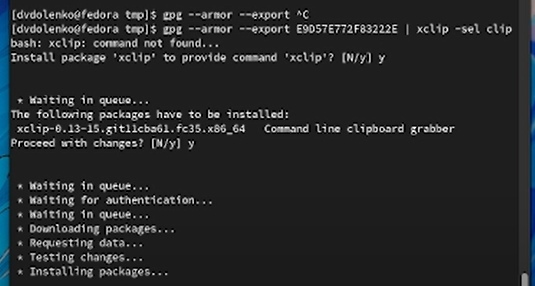


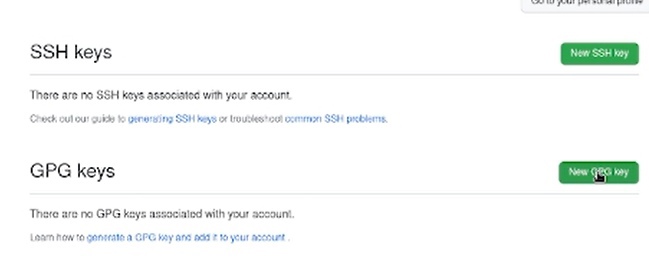
Генерирую ключ pgp:

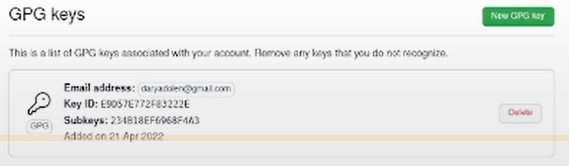


Добавляю ключ pgp в github:



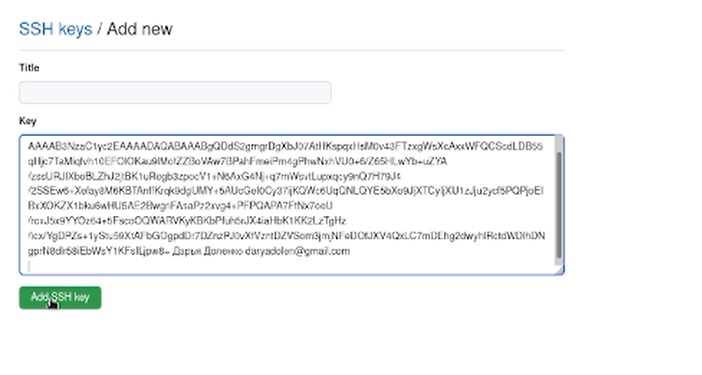




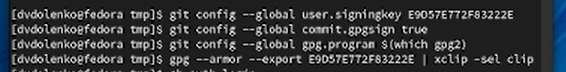


Добавляю ключ ssh в github аналогичным образом:

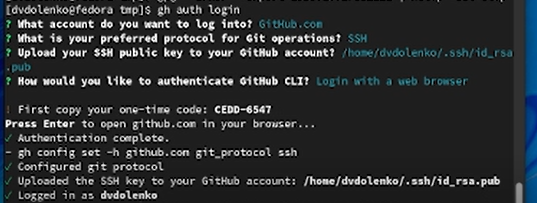




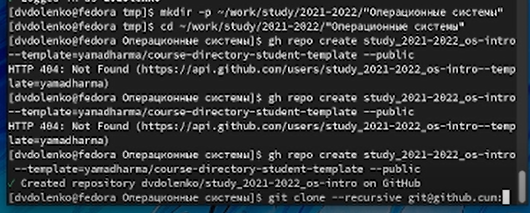
Настраиваю автоматические подписи коммитов git:



Настраиваю gh:



Создаю репозиторий курса на основе рабочего пространства:



**Вывод:** в ходе выполнения данной лабораторно работы я изучила идеологию и применение средств контроля версий, а также освоила умение по работе с git.

**Контрольные вопросы:**

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

Система контроля версия – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определенным старым версиям этих файлов. Мы будем использовать исходные коды программ, но под версионный контроль можно поместить файлы большинства типов. Так, например, веб-дизайнеры могут хранить каждую версию изображения, эта система позволяет возвращать отдельные файлы к прежнему виду.

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

Хранилище – это система, которая обеспечивает хранение всех существовавших вариантов файлов.

Commit – это фиксация изменений.

История – список предыдущих ревизий.

Рабочая копия – копия другой ветки.

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

Системы контроля версий: существуют централизованные – в ним один репозиторий, в него собираются изменения со всех рабочих копий разработчиков, и децентрализованные – имеют много репозиториев, могут обмениваться изменениями между собой. Централизированная система контроля версий Subversion и децентрализированная система контроля версий Mercurial.

1. *Опишите действия с VCS при единоличной работе с хранилищем.*

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документов из хранилища; обычно создается локальная копия документа. Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остается в хранилище и может быть оттуда получена в любое время. Сервер моет использовать так называемую дельта-компрессию – способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объем хранимых данных. Так как обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять ее целиком, заменяя в хранилище последнюю ранее сохраненную версия на разницу между этой и последней версией

1. *Опишите порядок работы с общим хранилищем VCS.*

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документов из хранилища; обычно создается локальная копия документа. Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остается в хранилище и может быть оттуда получена в любое время. Сервер моет использовать так называемую дельта-компрессию – способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объем хранимых данных. Так как обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять ее целиком, заменяя в хранилище последнюю ранее сохраненную версия на разницу между этой и последней версией

1. *Каковы основные задачи, решаемые инструментальным средством git?*

Устанавливает единственную новую команду git. Все возможности предоставляются через подкоманды этой команды. Можно просмотреть краткую справку командой help. С помощью help topics можно узнать список доступных тем, по которым группируются некоторые идеи. Система контроля версий также отслеживает, кто произвел изменения.

Основными задачами являются создание ветки, их размещение, объединение и публикация; просмотр и фиксация изменений, сообщение из текстового редактора, удаление зафиксированных изменений, выборочная фиксация, игнорирование файлов.

1. *Назовите и дайте краткую характеристику командам git.*

По мере внесения изменений в проект рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта увеличивается. Это повышает риск возникновения конфликтных изменений (см. ниже). Поэтому удобно поддерживать рабочую копию в состоянии, максимально близком к текущей основной версией, для чего разработчик выполняет операцию обновления рабочей копии (update) насколько возможно часто (реальная частота обновлений определяется частотой внесения изменений, зависящей от активности разработки и числа разработчиков, а также временем, затрачиваемым на каждое обновление — если оно велико, разработчик вынужден ограничивать частоту обновлений, чтобы не терять время). Модификация проекта Разработчик модифицирует проект, изменяя входящие в него файлы в рабочей копии в соответствии с проектным заданием. Эта работа производится локально и не требует обращений к серверу VCS. Фиксация изменений Завершив 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). VCS может требовать от разработчика перед фиксацией обязательно выполнить обновление рабочей копии. При наличии в системе поддержки отложенных изменений (shelving) изменения могут быть переданы на сервер без фиксации. Если утверждённая политика работы в VCS это позволяет, то фиксация изменений может проводиться не ежедневно, а только по завершении работы над заданием; в этом случае до завершения работы все связанные с заданием изменения сохраняются только в локальной рабочей копии разработчика.

1. *Приведите примеры использования при работе с локальным и удалённым репозиториями.*

Мы создаем новую ветку выполнив git init в уже созданном каталоге: % mkdir tutorial % cd tutorial % ls -a ./ ../ % pwd /home/mbp/work/bzr.test/tutorial % % git init % ls -aF ./ ../ .git/ % Мы обычно обращаемся к веткам на нашем компьютере просто передав имя каталога содержащего ветку. bzr также поддерживает доступ к веткам через http и sftp, например: git log http://bazaar-vcs.org git // git.dev/ git log sftp://bazaarvcs.org/bzr/bzr.dev/ Установив для git плагины можно также осуществлять доступ к веткам с использованием rsync. Команда status показывает какие изменения были сделаны в рабочем каталоге с момента последней ревизии: % git status modified: foo bzr status скрывает неинтересные файлы, которые либо не менялись, либо игнорируются. Также команде status могут быть переданы необязательные имена файлов, или каталогов для проверки. Команда diff показывает изменения в тексте файлов в стандартном формате diff. Вывод этой команды может быть передан другим командам, таким как ”patch”, ”diffstat”, ”filterdiff” и ”colordiff”: % git diff === added file ’hello.txt’ --- hello.txt 1970-01-01 00:00:00 +0000 +++ hello.txt 2005-10-18 14:23:29 +00006.2. Указания к лабораторной работе 75 @@ -0,0 +1,1 @@ +hello world Команде commit можно передать сообщение описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. git commit -m "добавлен первый файл" Если вы передадите список имен файлов, или каталогов после команды commit, то будут зафиксированы только изменения для переданных объектов. Например: bzr commit -m "исправления документации" commit.py Если вы сделали какие-либо изменения и не хотите оставлять их, используйте команду revert, что бы вернутся к состоянию предыдущей ревизии. 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 git игнорировать их добавив их в файл .ignore в корне рабочего дерева. Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду ignored: % ignored config.h ./config.h configure.in~ \*~ log Команда bzr log показывает список предыдущих ревизий. Команда log --forward делает тоже самое, но в хронологическом порядке, показывая более поздние ревизии в конце может контролировать файлы и каталоги, отслеживая переименования и упрощая их последующее объединение: % mkdir src % echo ’int main() {}’ > src/simple.c % add src added src added src/simple.c % status added: src/ src/simple.c bzr remove удаляет файл из под контроля версий, но может и не удалять рабочую копию файла2. Это удобно, когда вы добавили не тот файл, или решили, что файл на самом деле не должен быть под контролем версий. % rm -r src % remove -v hello.txt ? hello.txt % status removed: hello.txt src/ src/simple.c unknown: hello.txt Часто вместо того что бы начинать свой собственный проект, выхотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой.Если две ветки разошлись (обе имеют уникальные изменения) тогда merge — это подходящая команда для использования. Объединение автоматически вычислит изменения, которые существуют на объединяемой ветке и отсутствуют в локальной ветке и попытается объединить их с локальной веткой. git merge URL.

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

Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой эта команда называется branch: Управление версиями git branch cd git.dev Эта команда копирует полную историю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки.

1. *Как и зачем можно игнорировать некоторые файлы при commit?*

Нет проблем если шаблон для игнорирования подходит для файла под контролем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют показываются 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 игнорирования". 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 дерева. Этот файл содержит список шаблонов файлов, по одному в каждой строчке. Обычное содержимое может быть таким: \*.o \*~ \*.tmp \*.py [ co ] Если шаблон содержит 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 соответствующих им шаблонов используйте команду git ignored : $ git ignored config.h ./config.h configure.in~ \*~ $