**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

28 Ноября 2024 г**. №** \_\_\_\_\_\_\_\_\_\_

Факультет Кибербезопасности

Кафедра Инфокоммуникационных систем

# Отчет по лабораторной работе №7

«Знакомство с системой управления версиями git»

**по дисциплине «Технологии и методы программирования»**

Выполнили студенты гр. ИКБ-32  
Кушалиев Карим Искандерович Цыплаков Никита Андреевич

Принял ст.преп.

Воронцова И. О.

**Цель лабораторной работы**

Познакомить студентов со структурой локального репозитория git и основными операциями, доступными в нем. Работа выполняется в режиме терминала. После выполнения очередного действия следует проверять состояние (статус) репозитория.

**Задание**  
1. Войти в систему со своим логином/паролем и открыть эмулятор терминала.

2. Создать в домашнем каталоге каталог localrepo и в нем два файла — first.txt и second.txt. Просмотреть и зафиксировать детальную информацию о каталоге localrepo (сделать копию экрана или вывести в файл и сохранить для отчета).

3. Перейти в каталог localrepo. Инициализировать в каталоге localrepo репозиторий git (команда git init). Просмотреть и зафиксировать детальную информацию о каталоге localrepo. Как изменился каталог после создания репозитория? Изучить изменения — вывести на экран содержимое каталога .git, проверить статус репозитория (git status).

4. Добавить (команда git add) оба файла для сохранения в репозитории в следующем коммите. Проверить статус репозитория. Найти в каталоге .git файл index (в .git не переходить!). Зафиксировать время сохранения файла index.

5. Зафиксировать (сохранить) изменения в репозитории (git commit). Вывести на экран информацию о коммите (git show). Проверить статус репозитория. Проверить время сохранения файла index и сравнить его с предыдущим значением. Сделайте вывод о роли файла index при выполнении фиксации изменений в репозитории и почему команда git add называется «добавление в индекс».

6. Создать новый файл third.txt и зафиксировать его в репозитории. Отредактировать third.txt и зафиксировать в репозитории без предварительной индексации (команда commit с ключом -a).

7. Внести изменения в каждый из трех файлов. Проверить статус репозитория.

8. Для первого файла отменить последние изменения (git checkout). Второй и третий файлы проиндексировать. Проверить статус.

9. Отказаться от сохранения изменений второго файла (git reset HEAD), а изменения в третьем — зафиксировать в репозитории. Проверить статус. Выполнить команду, которая уберет second.txt из статуса (см. п. 8).

10. Вернуть последний коммит для внесения дополнительных изменений: внести изменения в файл, добавить их в индекс, затем выполнить команду git commit - -amend, при этом изменить текст комментария. Проверить статус и убедиться, что предыдущая редакция коммита отменена.

11. Просмотреть историю изменений репозитория (git log) — подробную и краткую.

12. Просмотреть историю изменений (логи) отдельного (любого) файла. Сравнить полученный результат с результатом, полученным для всего репозитория.

13. Изменить первый и второй файл, первый добавить в индекс. Для каждого файла сравнить текущую версию (с изменениями) и последнюю зафиксированную версию, для этого подобрать вариант команды git diff, соответствующий состоянию файла.

14. Сохранить все изменения в репозитории.

15. Переименовать файл first.txt средствами операционной системы (команда mv) — f1.txt, second.txt – средствами репозитория (git mv)— f2.txt.

16. Сделать копию третьего файла средствами операционной системы — f3.txt (команда cp).

17. Создать файл fourth.txt и добавить его в индекс.

18. Создать файл fifth.txt (можно пустой), но в индекс не добавлять.

19. Вывести на экран статус репозитория а) подробный, б) краткий. Изучить обозначения, используемые краткой формой представления статуса репозитория. Определить, какие изменения надо добавить в индекс для последующей фиксации, индексировать их и зафиксировать в репозитории.

20. Настроить игнорирование некоторых файлов, для этого создать в рабочем каталоге файл .gitignore и добавить в него шаблоны (маски) для имен файлов, которые репозиторий не должен замечать (они не будут отображаться в статусе). Задать три варианта шаблона имен файлов: 1) с символом \*, 2) c [ ], 3) c ? (знак вопроса).

21. Создать 3 файла с именами, удовлетворяющими этим шаблонам. Объяснить, почему эти имена соответствуют конкретному шаблону. Проверить статус репозитория и оценить результаты игнорирования файлов.

22. Проверить статус репозитория; если есть незафиксированные изменения, зафиксировать.

23. Удалить файл f2.txt средствами файловой системы. Проверить статус. Восстановить файл из репозитория.

24. Удалить файл third.txt командой git rm. Проверить статус. Зафиксировать удаление файла.

25. Отредактировать и добавить в индекс файл fifth.txt. Затем отредактировать его еще раз и проверить статус. Какие изменения будут зафиксированы последующим коммитом? Что надо сделать, чтобы все изменения были зафиксированы?

26. Отредактировать файл fifth.txt еще раз, добавить в индекс и зафиксировать, а затем исключить файл из списка отслеживаемых репозиторием. После выполнения операции файл должен остаться в каталоге!

27. Изменить файл fourth.txt, добавить в индекс, а затем удалить файл, несмотря на незафиксированные изменения — для этого подобрать подходящий вариант команды git rm.

28. Подготовить отчет на основе полученных результатов и ответить на вопросы.

**Листинг программы**

Основные команды:

1. touch – создает новый файл.

2. rm – удаляет файлы или каталоги.

3. git init – инициализирует пустой репозиторий Git в текущем каталоге.

4. git rm – удаляет файл из рабочей директории и индексирует удаление для следующего коммита.

5. git status – показывает состояние рабочего дерева, включая изменения, которые еще не были закоммичены.

6. git show – отображает информацию о конкретном коммите или объекте Git.

7. git add – добавляет изменения в индекс (staging area), чтобы они могли быть включены в следующий коммит.

8. git checkout – переключается между ветками или восстанавливает рабочие файлы из индекса или другого коммита.

9. git reset HEAD – отменяет добавление изменений в индекс (перед последним коммитом).

10. git log – выводит историю коммитов проекта.

11. git diff – сравнивает изменения между двумя версиями файлов или между рабочим деревом и индексом.

12. git mv – переименовывает или перемещает файл, сохраняя при этом историю версий.Git rm --cached file– исключаем файл из отслеживания

В .gitignore:

# Игнорируем все файлы с расширением ".bak" в любом месте проекта

\*.bak

# Игнорируем файлы, начинающиеся с буквы "a", "b" или "c", за которой следует любое число цифр

[abc][0-9]\*.log

# Игнорируем файлы, содержащие одну произвольную букву между двумя фиксированными буквами "a" и "b"

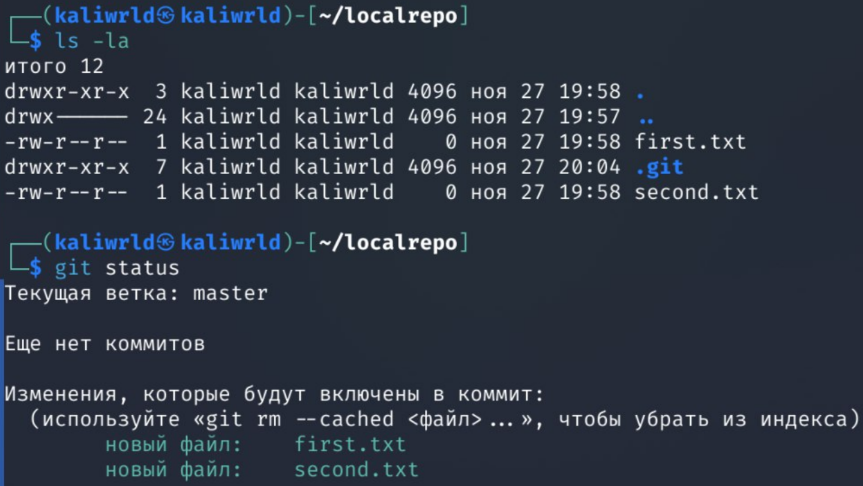
a?b.\*

# Игнорируем fifth.txt

Fifth.txt

**Ход работы**

Как изменился каталог после создания репозитория?



Сделайте вывод о роли файла index при выполнении фиксации изменений в репозитории и почему команда git add называется «добавление в индекс».

* Файл index. Его основное назначение заключается в том, чтобы отслеживать текущее состояние рабочего дерева перед фиксацией изменений. Когда вы вносите изменения в файлы в рабочем дереве, эти изменения еще не фиксируются в репозитории. Чтобы подготовить их к фиксации, используется команда git add. Эта команда добавляет изменённые файлы в индекс. Таким образом, индекс становится промежуточным этапом между рабочим деревом и коммитом.
* Роль индекса:

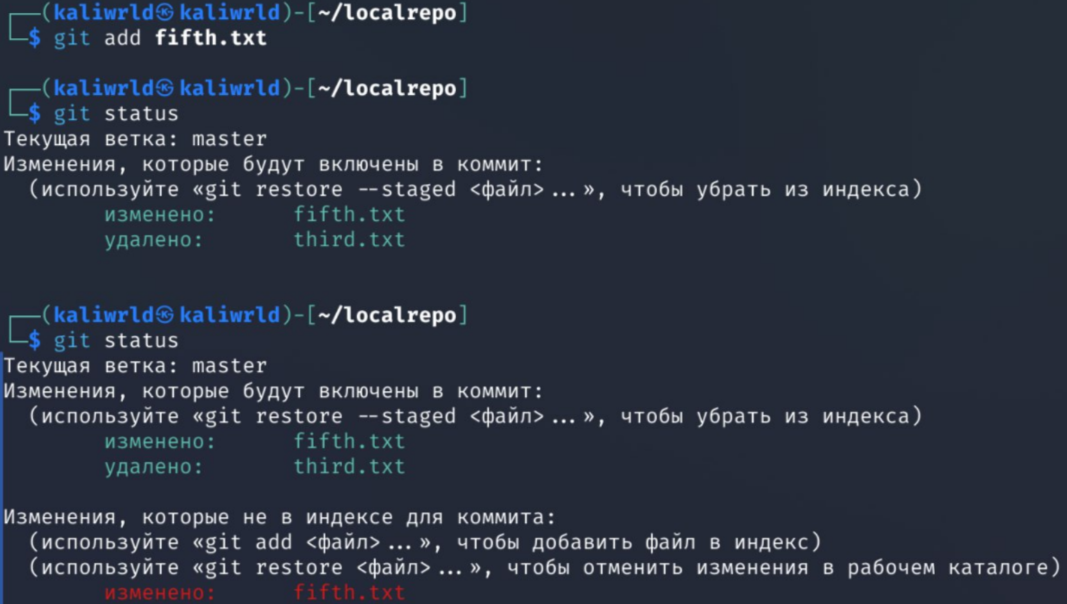
1. Отслеживание изменений: Индекс отслеживает, какие файлы были изменены и готовы к фиксации. Это позволяет вам контролировать, какие именно изменения будут включены в следующий коммит.

2. Разделение изменений: Вы можете добавлять отдельные файлы или части файлов в индекс, а затем создавать коммиты поэтапно. Например, вы можете добавить часть изменений одного файла, создать коммит, а затем добавить оставшиеся изменения и сделать ещё один коммит.

3. Объединение изменений: Если вы внесли несколько изменений в разные файлы, но хотите объединить их в одном коммите, то сначала добавляете все нужные файлы в индекс командой git add, а затем создаёте коммит командой git commit.

Команда git add означает «добавить файл в индекс», потому что она помещает изменённый файл в область подготовки (staging area), где он ожидает дальнейшей обработки (фиксации). Только после того, как файл был добавлен в индекс, он может быть зафиксирован в репозитории с помощью команды git commit.

Какие изменения будут зафиксированы последующим коммитом? Что надо сделать, чтобы все изменения были зафиксированы?



Что произойдет при следующем коммите?

Все изменения, которые были внесены в файл fifth.txt и добавлены в индекс (git add), будут зафиксированы следующим коммитом. То есть оба этапа редактирования (First change и Second change) войдут в этот коммит.

Как убедиться, что все изменения будут зафиксированы?

git commit -m "изменение fifth.txt"

**Вывод**

Были освоены основные принципы работы с структурой локального репозитория git и операциями в нем. Работа выполнялась в режиме терминала.