

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.2
дисциплины «Основы кроссплатформенного программирования»

Вариант ____

Выполнила:
Маньшина Дарья Алексеевна
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. тех. наук,
доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

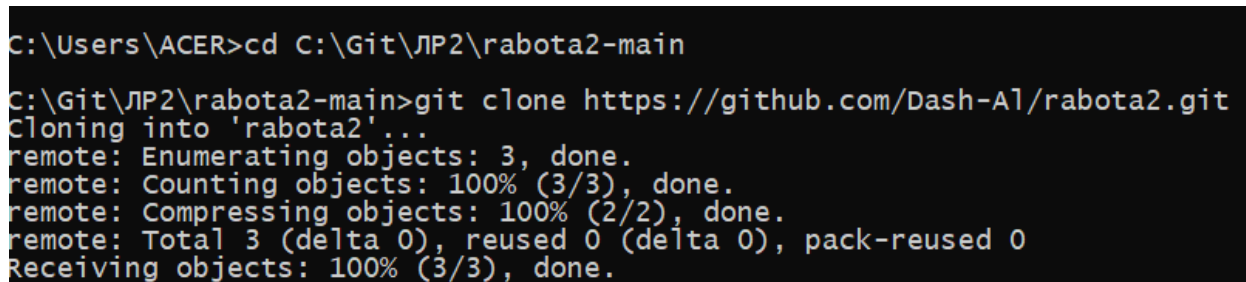
Ставрополь, 2023 г.

Тема: исследование возможностей Git для работы с локальными репозиториями

Цель: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Ход работы:

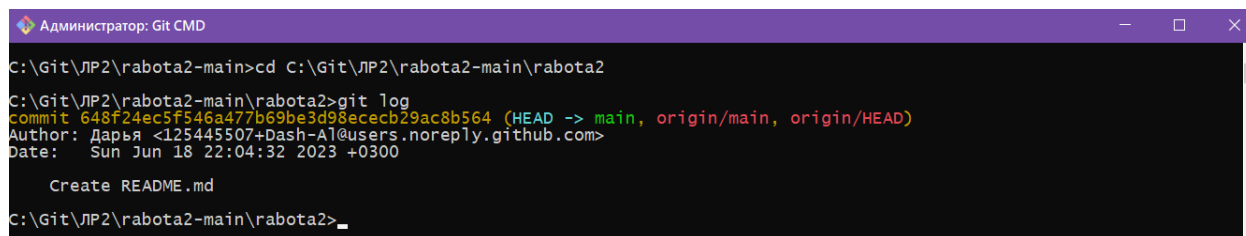
1. Создаем новый репозиторий и клонируем его.



```
C:\Users\ACER>cd C:\Git\ЛР2\rabota2-main
C:\Git\ЛР2\rabota2-main>git clone https://github.com/Dash-A1/rabota2.git
Cloning into 'rabota2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 1 – Клонирование.

2. Проверка работы «git log».

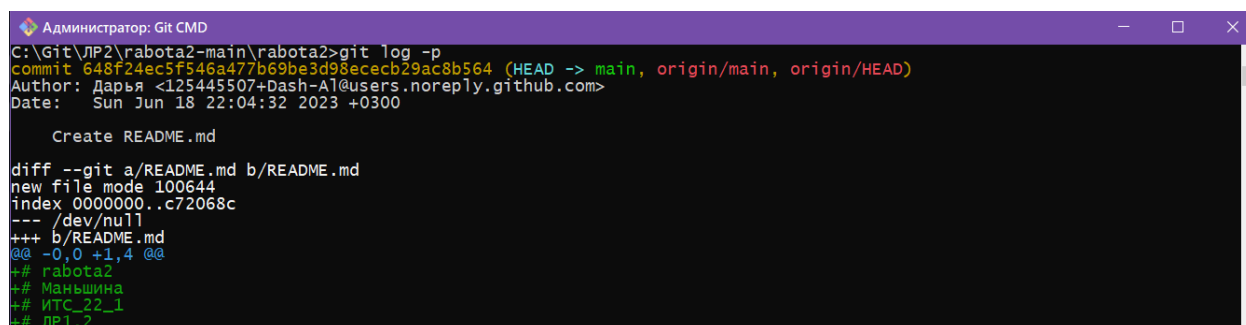


```
Администратор: Git CMD
C:\Git\ЛР2\rabota2-main>cd C:\Git\ЛР2\rabota2-main\rabota2
C:\Git\ЛР2\rabota2-main\rabota2>git log
commit 648f24ec5f546a477b69be3d98ecec29ac8b564 (HEAD -> main, origin/main, origin/HEAD)
Author: Дарья <125445507+Dash-A1@users.noreply.github.com>
Date: Sun Jun 18 22:04:32 2023 +0300

    Create README.md
C:\Git\ЛР2\rabota2-main\rabota2>
```

Рисунок 2 – Итог работы «git log».

3. Проверка работы команды «git log -p».



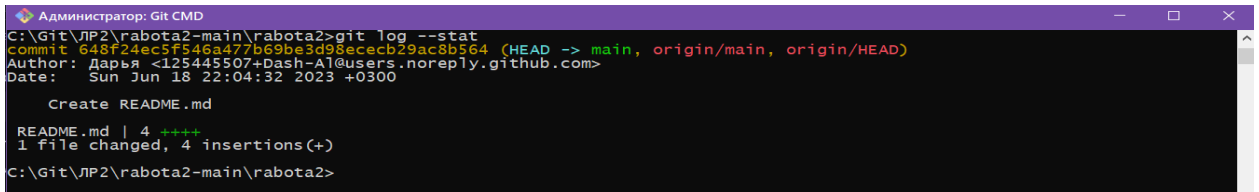
```
Администратор: Git CMD
C:\Git\ЛР2\rabota2-main\rabota2>git log -p
commit 648f24ec5f546a477b69be3d98ecec29ac8b564 (HEAD -> main, origin/main, origin/HEAD)
Author: Дарья <125445507+Dash-A1@users.noreply.github.com>
Date: Sun Jun 18 22:04:32 2023 +0300

    Create README.md

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..c72068c
--- /dev/null
+++ b/README.md
@@ -0,0 +1,4 @@
+# работа2
+# Маньшина
+# ИТС_22_1
+# ЛР1.2
```

Рисунок 3 – Итог работы «git log -p».

4. Проверка работы команды «git log –stat».



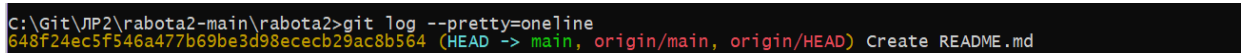
```
Администратор: Git CMD
C:\Git\ЛР2\rabota2-main\rabota2>git log --stat
commit 648f24ec5f546a477b69be3d98ecb29ac8b564 (HEAD -> main, origin/main, origin/HEAD)
Author: Дарья <125445507+Dash-A1@users.noreply.github.com>
Date: Sun Jun 18 22:04:32 2023 +0300

    Create README.md

 README.md | 4 +++++
 1 file changed, 4 insertions(+)
```

Рисунок 4 – Итог работы «git log --stat».

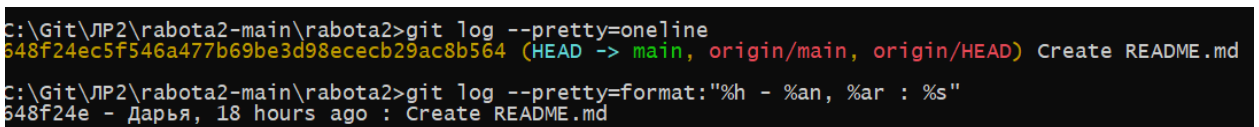
5. Проверка работы команды «git log –pretty=oneline».



```
C:\Git\ЛР2\rabota2-main\rabota2>git log --pretty=oneline
648f24ec5f546a477b69be3d98ecb29ac8b564 (HEAD -> main, origin/main, origin/HEAD) Create README.md
```

Рисунок 5 – Итог работы «git log –pretty=oneline».

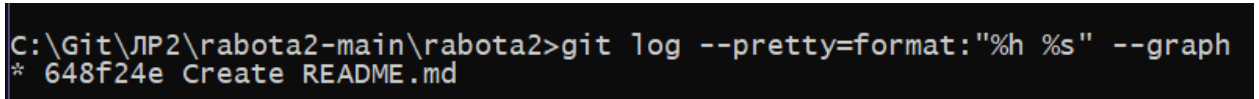
6. Проверка работы команды «git log --pretty=format:"%h - %an, %ar : %s"».



```
C:\Git\ЛР2\rabota2-main\rabota2>git log --pretty=oneline
648f24ec5f546a477b69be3d98ecb29ac8b564 (HEAD -> main, origin/main, origin/HEAD) Create README.md
C:\Git\ЛР2\rabota2-main\rabota2>git log --pretty=format:"%h - %an, %ar : %s"
648f24e - Дарья, 18 hours ago : Create README.md
```

Рисунок 6 – Итог работы «git log --pretty=format:"%h - %an, %ar : %s"».

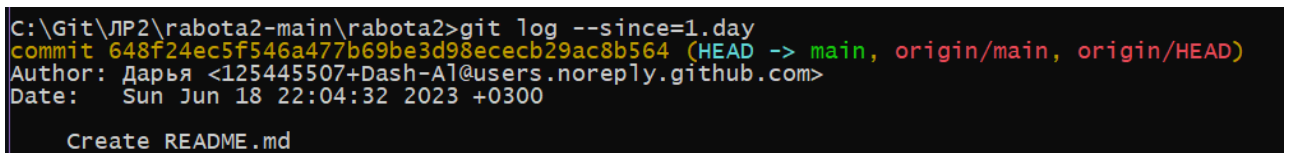
7. Проверка работы команды «git log --pretty=format:"%h %s" --graph».



```
C:\Git\ЛР2\rabota2-main\rabota2>git log --pretty=format:"%h %s" --graph
* 648f24e Create README.md
```

Рисунок 7 – Итог работы «git log --pretty=format:"%h %s" --graph».

8. Работа с командой «git log --since=1.day»



```
C:\Git\ЛР2\rabota2-main\rabota2>git log --since=1.day
commit 648f24ec5f546a477b69be3d98ecb29ac8b564 (HEAD -> main, origin/main, origin/HEAD)
Author: Дарья <125445507+Dash-A1@users.noreply.github.com>
Date: Sun Jun 18 22:04:32 2023 +0300

    Create README.md
```

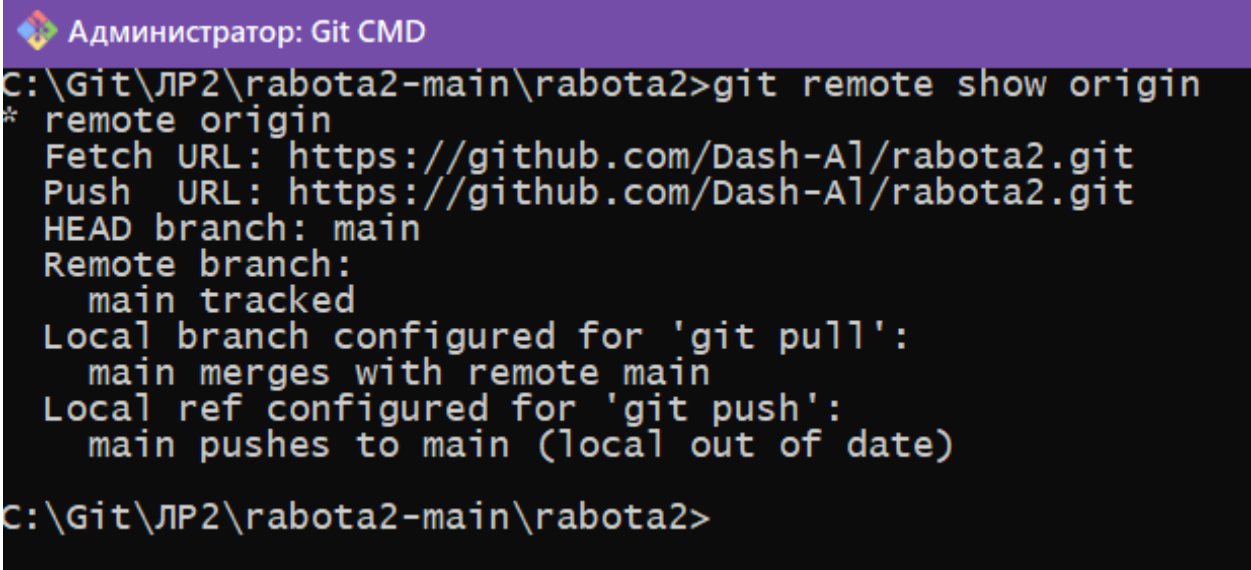
Рисунок 8 – Итог работы «git log --since=1.day»

9. Работа с командой «git remote -v»

```
C:\Git\ЛР2\rabota2-main\rabota2>git remote -v
origin  https://github.com/Dash-A1/rabota2.git (fetch)
origin  https://github.com/Dash-A1/rabota2.git (push)
```

Рисунок 9 – Итог работы «git remote -v»

10. Работа с командой «git remote show origin».



Администратор: Git CMD

```
C:\Git\ЛР2\rabota2-main\rabota2>git remote show origin
* remote origin
Fetch URL: https://github.com/Dash-A1/rabota2.git
Push URL: https://github.com/Dash-A1/rabota2.git
HEAD branch: main
Remote branch:
  main tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (local out of date)

C:\Git\ЛР2\rabota2-main\rabota2>
```

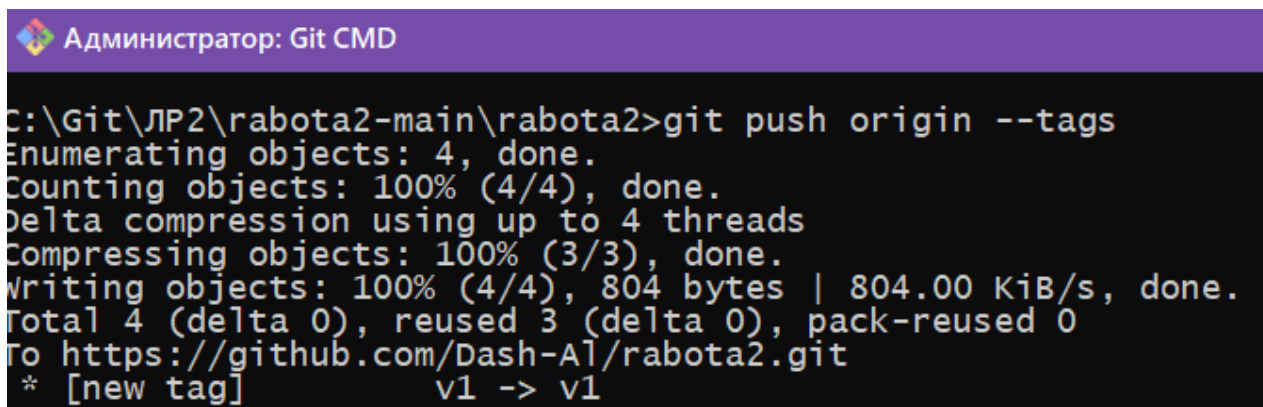
Рисунок 10 – Итог работы «git remote show origin».

11. Как и большинство СКВ, Git имеет возможность помечать определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами.

```
C:\Git\ЛР2\rabota2-main\rabota2>git tag -a v1 -m "v1"
C:\Git\ЛР2\rabota2-main\rabota2>git tag
v1
```

Рисунок 11 – Добавление тегов и просмотр списков тегов.

12. Чтобы отправить теги на удалённый сервер, нужно воспользоваться командой «git push origin --tags».

A screenshot of a Windows command prompt window titled "Администратор: Git CMD". The command prompt shows the execution of the command "git push origin --tags" in a directory "C:\Git\ЛР2\rabota2-main\rabota2". The output shows the process of enumerating, counting, compressing, and writing objects, resulting in a successful push to "https://github.com/Dash-A1/rabota2.git" with a new tag "v1" created from "v1".

```
Администратор: Git CMD
C:\Git\ЛР2\rabota2-main\rabota2>git push origin --tags
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 804 bytes | 804.00 KiB/s, done.
Total 4 (delta 0), reused 3 (delta 0), pack-reused 0
To https://github.com/Dash-A1/rabota2.git
 * [new tag]          v1 -> v1
```

Рисунок 12 – Команда «git push origin --tags».

Ответы на контрольные вопросы:

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Ответ: если нам понадобится посмотреть, что было сделано — историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда git log.

Опция и описание: **-p** показывает патч для каждого коммита; **--stat** показывает статистику измененных файлов для каждого коммита; **--shortstat** отображает только строку с количеством изменений/вставок/удалений для команды **--stat**; **--name-only** показывает список измененных файлов после информации о коммите; **--name-status** показывает список файлов, которые добавлены/изменены/удалены; **--abbrev-commit** показывает только несколько символов SHA-1 чек-суммы вместо всех 40; **--relative-date** отображает дату в относительном формате вместо стандартного формата даты; **--graph** отображает ASCII граф с ветвлениями и историей слияний; **--pretty** показывает коммиты в альтернативном формате. Возможные варианты опций: **oneline**, **short**, **full**, **fuller** и **format** (с помощью последней можно указать свой формат); **--oneline** сокращение для одновременного использования опций **--pretty=oneline --abbrev-commit**.

2. Как ограничить вывод при просмотре истории коммитов?

Ответ: при помощи опции `-<n>`, где `n` — это любое натуральное число и представляет собой `n` последних коммитов. Опции для ограничения вывода по времени, такие как `--since` и `--until`. опция `-s`, которая принимает аргумент в виде строки и показывает только те коммиты, в которых изменение в коде повлекло за собой добавление или удаление этой строки.

3. Как внести изменения в уже сделанный коммит?

Ответ: Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр `--amend` : `git commit --amend`

4. Как отменить индексацию файла в Git?

Ответ: используйте `git reset HEAD<file>...` для исключения из индекса.

5. Как отменить изменения в файле?

Ответ: `$ git checkout --`

6. Что такое удаленный репозиторий Git?

Ответ: Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Ответ: для просмотра списка удаленных репозиториях, которые сконфигурированы в данный момент для данного локального репозитория, используется команда: `git remote -v`. Ключ `-v` означает *verbose* и используется, чтобы выводить подробную информацию.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Ответ: для того, чтобы добавить удалённый репозиторий и присвоить ему имя (`shortname`), просто выполните команду `git remote add <shortname> <url>`

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Ответ: `$ git fetch [remote-name]`

Выполнение `git pull`, как правило, извлекает (`fetch`) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (`merge`) их с кодом, над которым вы в данный момент работаете.

10. Как выполнить просмотр удаленного репозитория?

Ответ: для того, чтобы просмотреть список настроенных удалённых репозиториях, вы можете запустить команду `git remote`. Она выведет названия доступных удалённых репозиториях. Если вы клонировали репозиторий, то увидите, как минимум `origin` — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование

11. Каково назначение тэгов Git?

Ответ: возможность пометить определённые моменты в истории как важные.

12. Как осуществляется работа с тэгами Git?

Ответ: просмотреть список имеющихся тегов в Git можно набрав команду `git tag` (параметры `-l` и `--list` опциональны); создание аннотированного тега в Git — указать `-a` при выполнении команды `tag`; с помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом.

13. Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага?

Ответ: `git fetch --prune` — лучшая утилита для очистки устаревших веток. Она подключается к удаленному репозиторию с общим доступом и получает все ссылки на удаленные ветки. Затем ссылки, которые больше не используются в удаленном репозиторию, обрезаются.

Вывод: в ходе лабораторной работы исследовала базовые возможности системы контроля версий Git для работы с локальными репозиториями.