

Харьковский национальный университет им. В.Н. Каразина

Факультет компьютерных наук

Лабораторная работа №1

По учебной дисциплине

**Математические методы и технологии тестирования и верификации
программного обеспечения**

«Изучение системы контроля версии на примере Git»

Выполнил:

Студент группы КС-22

Филатов Виталий Витальевич

Проверил:

Доц. Малахов Сергей Витальевич

Тема: Изучение системы контроля версии на примере Git.

Цель работы: Цель – получить базовые знания о том, что такое Git и чем он отличается от централизованных систем контроля версий. Также получить рабочую версию Git на компьютере, настроенную и персонализированную.

1. Установка Git

Установочный файл можно скачать с официального сайта <https://git-scm.com>. После запуска открывается окно установки Git (Рисунок 1.1).

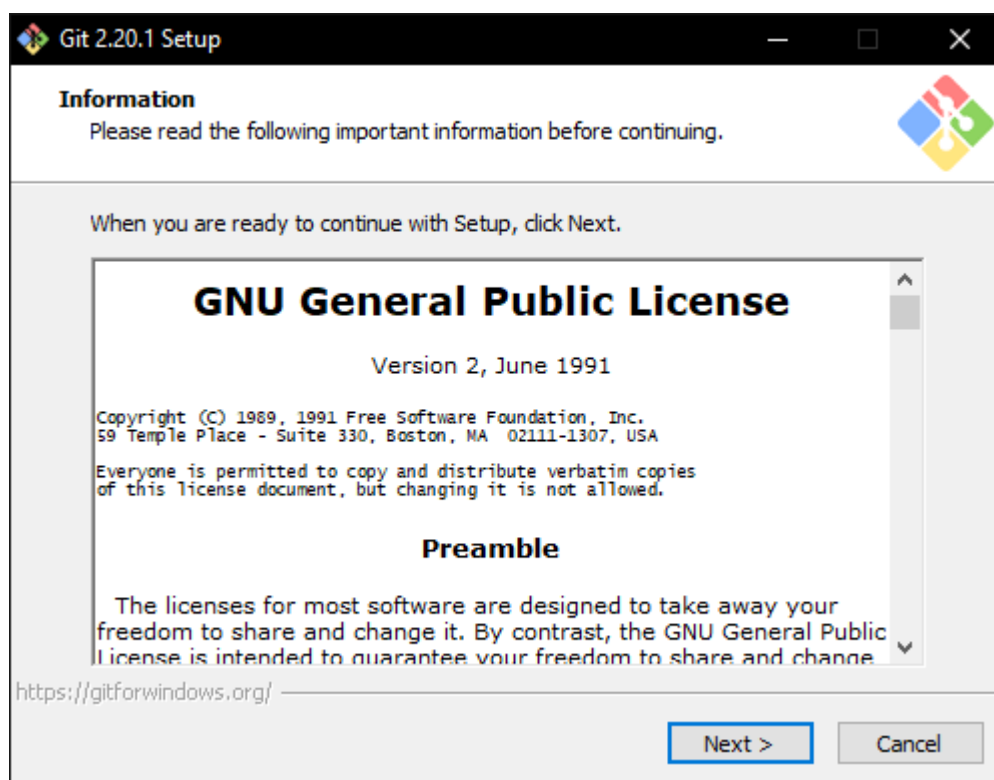


Рисунок 1.1 – окно установки Git

Нажимаем next, до полной установки. После установки запускаться консоль (Рисунок 1.2)

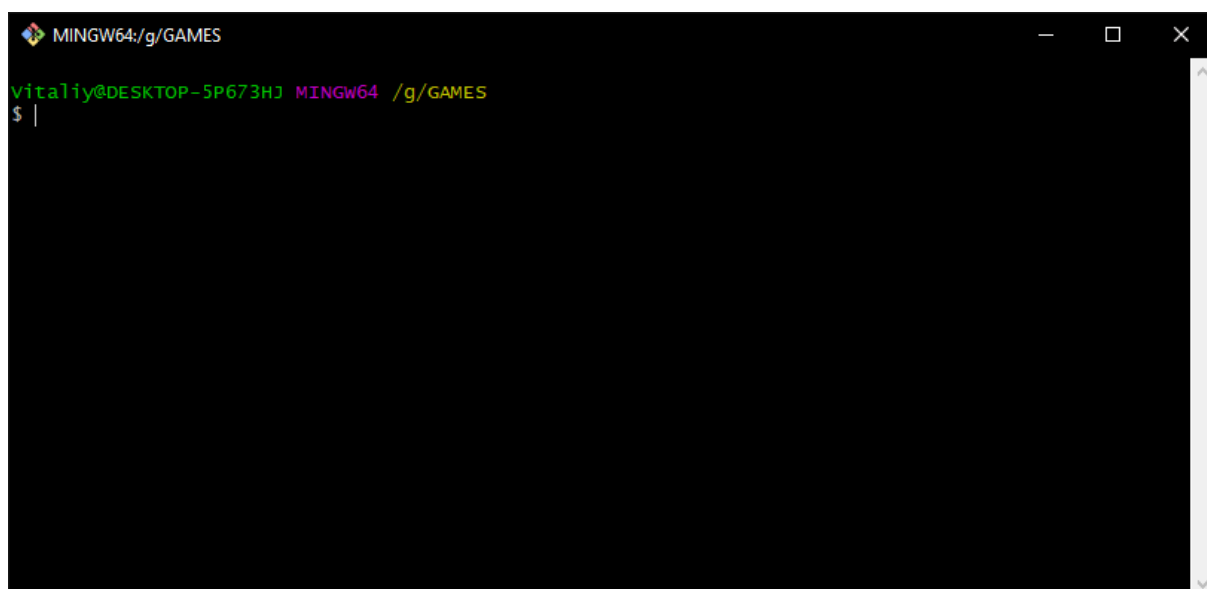


Рисунок 1.2 – окно Git Bash

2. Первоначальная настройка Git

Настройку нужно сделать только один раз. При обновлении версии настройки сохраняются. Но при необходимости, вы можете поменять настройки в любой момент (Рисунок 2.1, Рисунок 2.2).

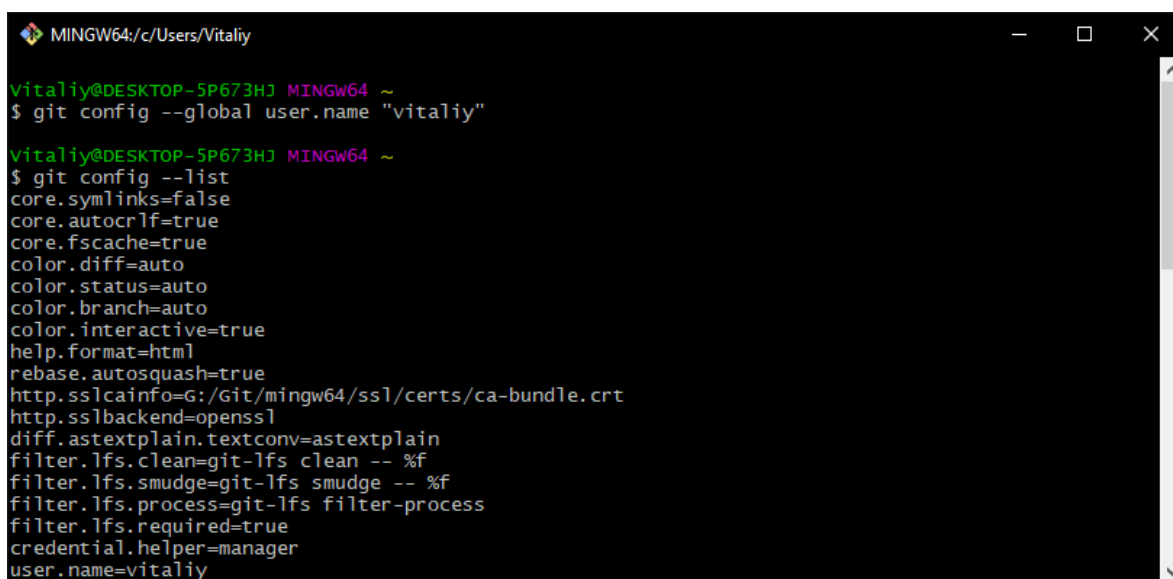


Рисунок 2.1 – установка имени пользователя

```

MINGW64:/c/Users/Vitaliy
vitaliy@DESKTOP-5P673HJ MINGW64 ~
$ git config --global user.email "user@mail.ru"

vitaliy@DESKTOP-5P673HJ MINGW64 ~
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=G:/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.name=vitaliy

```

Рисунок 2.2 – установка почты пользователя

3. Создание коммитов

Изначально надо создать локальный Git репозиторий с помощью команды «*git init*» (Рисунок 3.1).

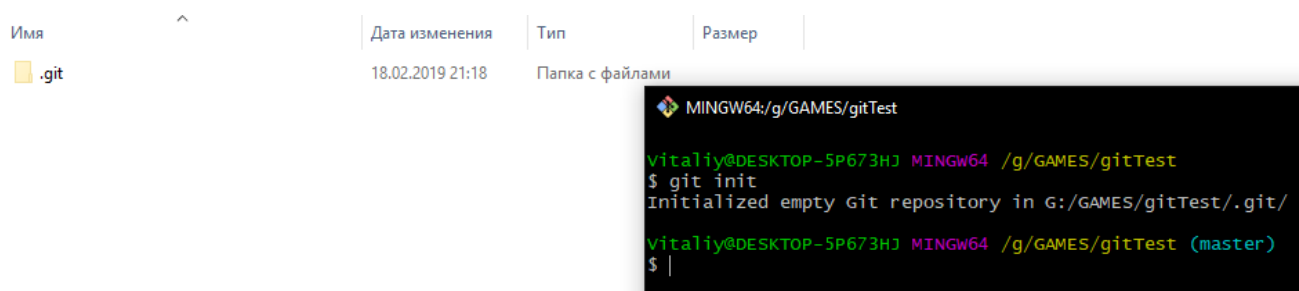


Рисунок 3.1 – инициализация локального Git репозитория

Далее необходимо добавить файл в наш Git репозиторий (Рисунок 3.2).

Имя	Дата изменения	Тип	Размер
.git	18.02.2019 17:46	Папка с файлами	
text.txt	18.02.2019 17:46	Файл "ТХТ"	0 КБ

Рисунок 3.2 – добавление файла в репозиторий

Далее необходимо зарегистрировать наш файл в репозитории с помощью команды «*git add .*» (Рисунок 3.3).

```

MINGW64:/g/GAMES/gitTest

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git add .
vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)

```

Рисунок 3.3 – добавление файлов в репозиторий

Теперь остается только сделать коммит командой «*git commit -m "Message"*» (Рисунок 3.4).

```

MINGW64:/g/GAMES/gitTest

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git add .

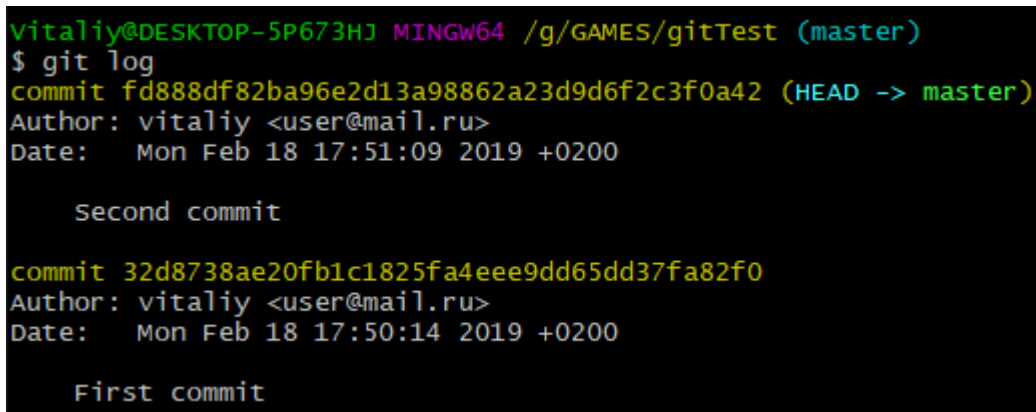
vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git commit -m "First commit"
[master (root-commit) 32d8738] First commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 text.txt

```

Рисунок 3.4 – создание коммита

4. Просмотр истории изменений в файлах

Для просмотра истории коммитов используется команда «*git log*» (Рисунок 4.1).



```
vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git log
commit fd88df82ba96e2d13a98862a23d9d6f2c3f0a42 (HEAD -> master)
Author: vitaliy <user@mail.ru>
Date:   Mon Feb 18 17:51:09 2019 +0200

    Second commit

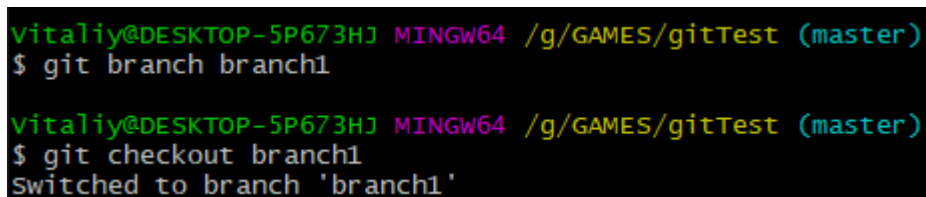
commit 32d8738ae20fb1c1825fa4eee9dd65dd37fa82f0
Author: vitaliy <user@mail.ru>
Date:   Mon Feb 18 17:50:14 2019 +0200

    First commit
```

Рисунок 4.1 – история коммитов, выведенная командой *git commit*

5. Создание новых веток и переключение между ними

Для создания новых веток используется команда «*git branch название_ветки*». Для переключения между ветками используется команда «*git checkout имя_ветки*» (Рисунок 5.1).



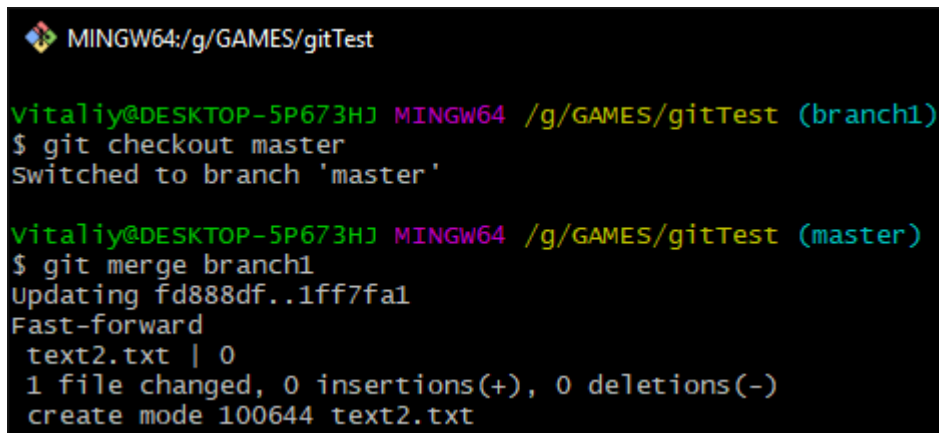
```
vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git branch branch1

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git checkout branch1
Switched to branch 'branch1'
```

Рисунок 5.1 – создание новой ветки и переключение на неё

6. Слияние веток

Для слияния веток необходимо переключиться на ветку, которую надо обновить и выполнить команду «*git merge название_ветки*». (Рисунок 6.1)



```
MINGW64:/g/GAMES/gitTest

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (branch1)
$ git checkout master
Switched to branch 'master'

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git merge branch1
Updating fd88df..1ff7fa1
Fast-forward
 text2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 text2.txt
```

Рисунок 6.1 – слияние ветки branch1 с веткой master

7. Отправка данных на удаленный репозиторий

Удаленному репозиторию можно присвоить локальное имя (Рисунок 7.1).

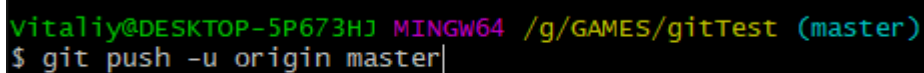


```
MINGW64:/g/GAMES/gitTest

vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git remote add origin https://github.com/Gashor/study.git
```

Рисунок 7.1 – присвоение имени удаленному репозиторию

Для отправки данных на удаленный репозиторий используется команда «*git push -u имя_репозитория имя_нашей_ветки*» (Рисунок 7.2).



```
vitaliy@DESKTOP-5P673HJ MINGW64 /g/GAMES/gitTest (master)
$ git push -u origin master|
```

Рисунок 7.2 – отправки данных с ветки на удаленный репозиторий

После выполнения команды файлы будут находиться в удаленном репозитории, откуда их потом можно будет в дальнейшем скачать и изменить (Рисунок 7.3).

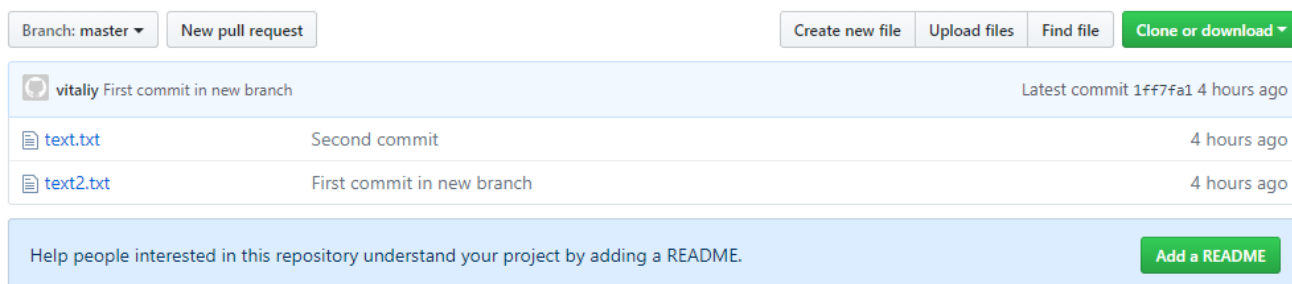


Рисунок 7.3 – данные находятся в удаленном Git репозитории

ВЫВОДЫ

В ходе лабораторной работы была установлена система контроля версий Git. Я научился добавлять файлы в репозиторий, создавать новые ветки, переключаться между ними, выполнять слияние веток, а также отправлять данные на удаленный Git репозиторий.