

Основы программной инженерии (ПОИТ)
Технологии разработки программного обеспечения (ИСИТ)

Системы контроля версий


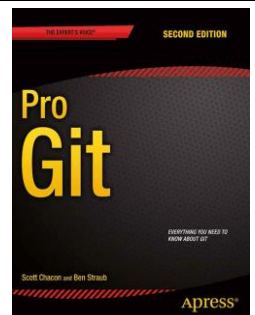
План лекции:

- назначение, разновидности систем контроля версий;
- система контроля версий Git;
- установка и настройка Git;
- три состояния файлов в Git;
- основные команды в Git;
- ветвления в Git;
- создание, слияние веток в Git;
- конфликты при слиянии веток в Git.

1. Системы контроля версий

Система управления **версиями** (от англ. *VersionControl System, VCS* или *Revision Control System, RCS*) – программное обеспечение для облегчения работы с изменяющейся информацией и разработки проекта совместно с коллегами.

Литература:

| | |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
|  | Выложена на diskstation.belstu.by |
|  | Доступна по ссылке: https://git-scm.com/book/ru/v2 |

Назначение

| | |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назначение систем контроля версий | <ul style="list-style-type: none">✓ <i>автоматическое создание архива (бэкап) для синхронизации кодовой базы;</i>✓ <i>отслеживание изменений (кто, когда и зачем сделал изменения);</i>✓ <i>совместная работа над одним и тем же проектом;</i>✓ <i>отслеживание ошибок (Bug трекинг-система).</i> |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

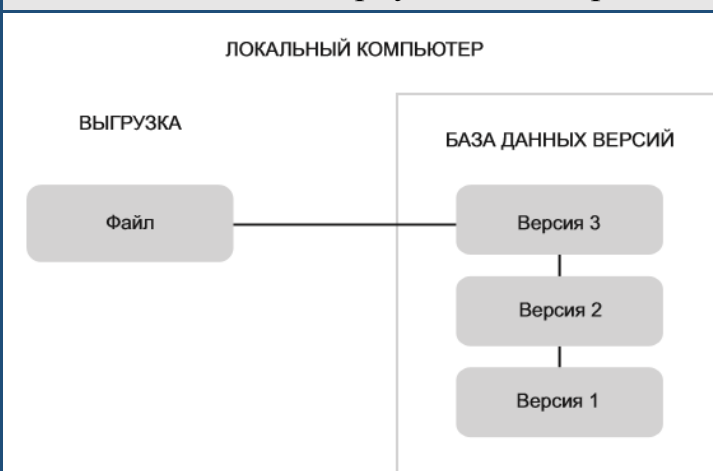
Разновидности

| | |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Разновидности систем контроля версий | <ul style="list-style-type: none">✓ <i>локальные системы контроля версий (Version Control System, VCS, Revision Control System, RCS);</i>✓ <i>централизованные системы контроля версий (Centralized Version Control System, CVCS);</i>✓ <i>распределенные системы контроля версий (Distributed Version Control System, DVCS).</i> |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Локальные системы контроля версий

Revision Control System, RCS

записывает историю изменения файла или набора файлов, чтобы в будущем была возможность вернуться к конкретной версии.



Репозиторий, хранилище — место, где хранятся и поддерживаются какие-либо данные.

Изменения сохраняются в виде наборов **патчей** (обновления, изменения ПО), где каждый патч помечается меткой даты-времени.

Централизованные системы контроля версий

Centralized Version Control System, CVCS

с единым сервером, содержащим все версии файлов, и набором клиентов, получающих файлы с сервера, что позволяет решить проблему взаимодействия с другими разработчиками.

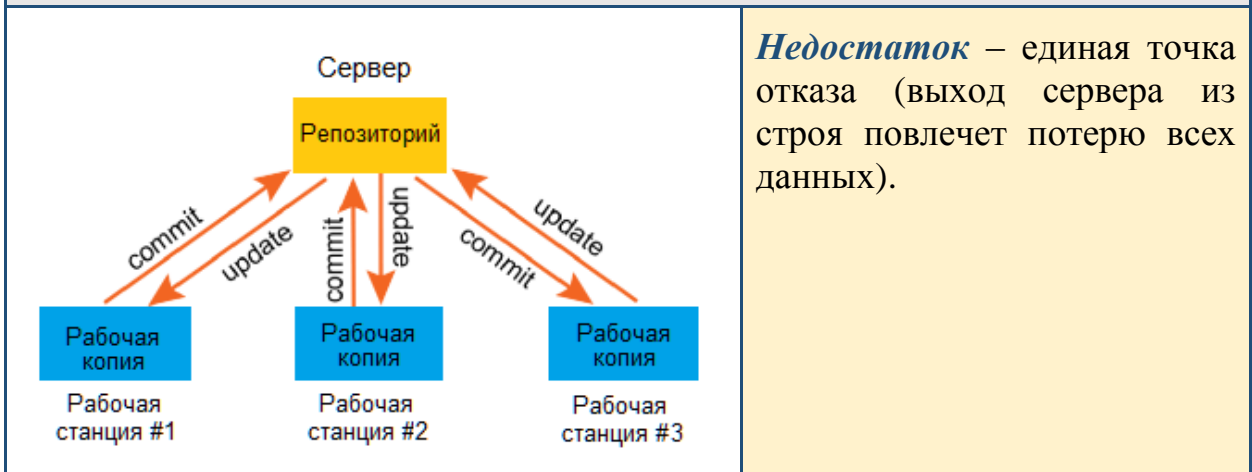
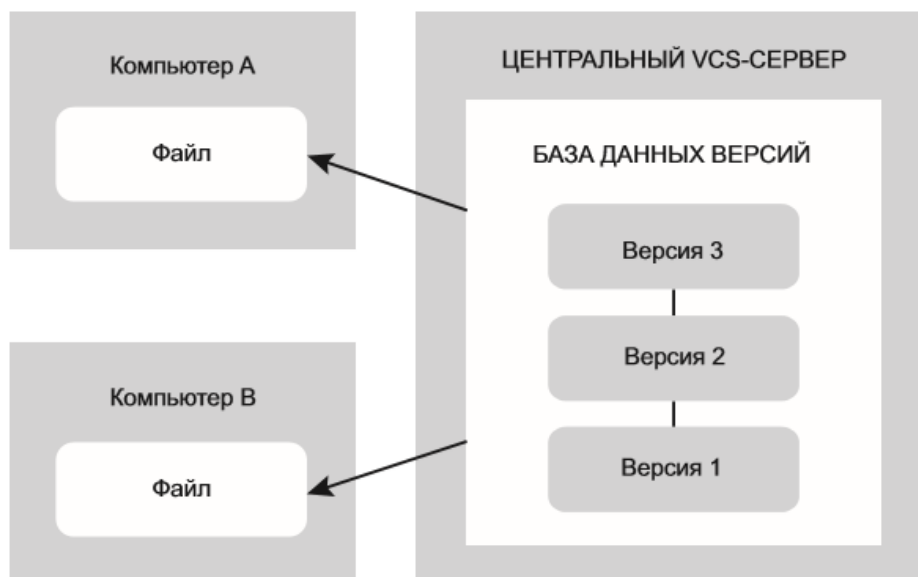


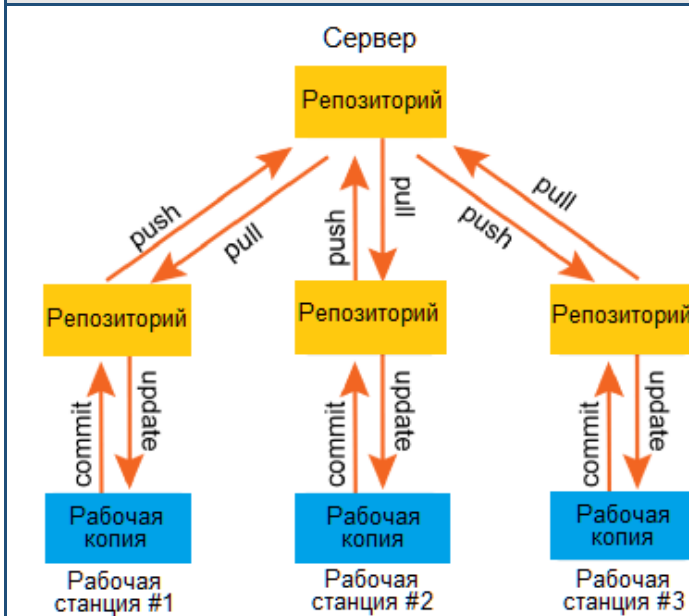
Схема централизованной системы контроля версий:



Распределенные системы контроля версий




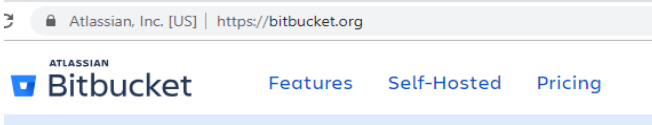
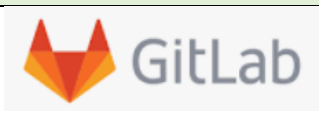
Distributed Version Control System, DVCS

клиенты полностью копируют репозиторий (у каждого клиента есть копия всего исходного кода и внесённых изменений).



Преимущество — позволяют одновременно взаимодействовать с несколькими удалёнными репозиториями, таким образом, обеспечивается возможность параллельной работы над несколькими проектами.

Сравнение решений

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Система одновременных версий (CVS) (80е) | централизованная система управления версиями, популярная в 1990-е – начале 2000-х годов. Хранит историю изменений определённого набора файлов. Создана Диком Груном в 1986 г |
| SVN (Subversion) |  |
| свободная централизованная система управления версиями. <i>Дата запуска:</i> 2000 г. <i>Язык программирования:</i> Си | |
| Git |  |
| распределённая система управления версиями. <i>Дата запуска:</i> 2005 г. <i>Язык программирования:</i> Си, командная оболочка UNIX, Perl, Tcl, Python и C++ <i>Разработчик:</i> Линус Торвалдс | |
| Mercurial |  |
| кроссплатформенная распределённая система управления версиями, разработанная для эффективной работы с очень большими репозиториями кода. Является консольной программой. <i>Дата запуска:</i> 2005 г. <i>Язык программирования:</i> Python, Си, Rust <i>Разработчик:</i> Мэттом Макколлом | |
| BITBUCKET |  |
| веб-сервис для хостинга проектов и их совместной разработки, основанный на системах контроля версий Mercurial и Git <i>Дата запуска:</i> 2008 г. <i>Язык программирования:</i> Python <i>Разработчик:</i> | |
| GitLab |  |
| веб-приложение с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок <i>Дата запуска:</i> 2011 г. <i>Язык программирования:</i> Ruby, Go | |

Система одновременных версий (CVS)

Преимущества:

- ✓ Испытанная временем технология, которая удерживается на рынке десятки лет.

Недостатки:

- ✓ Переименование или перемещение файлов не отражается в истории
- ✓ Риски безопасности, связанные с символическими ссылками на файлы
- ✓ Нет поддержки атомарных операций, что может привести к повреждению кода
- ✓ Операции с ветками программного кода дорогостоящие, так как эта система контроля не предназначена для долгосрочных проектов с ветками кода

Преимущества SVN:

- ✓ Система на основе CVS
- ✓ Допускает атомарные операции
- ✓ Операции с ветвлением кода менее затратны
- ✓ Широкий выбор плагинов IDE
- ✓ Не использует пиринговую модель

Недостатки:

- ✓ Сохраняются ошибки, связанные с переименованием файлов и папок
- ✓ Неудовлетворительный набор команд для работы с репозиторием
- ✓ Сравнительно небольшая скорость

Преимущества Git:

- ✓ Значительное увеличение быстродействия
- ✓ Дешевые операции с ветками кода
- ✓ Полная история разработки доступна оффлайн
- ✓ Распределенная, пиринговая модель

Недостатки:

- ✓ Высокий порог вхождения для тех, кто ранее использовал SVN
- ✓ Ограниченная поддержка Windows (по сравнению с Linux)

Преимущества Mercurial:

- ✓ По сравнению с Git легче в освоении
- ✓ Подробная документация
- ✓ Распределенная модель системы контроля версий

Недостатки:

- ✓ Нет возможности слияния двух родительских веток
- ✓ Использование плагинов, а не скриптов
- ✓ Меньше возможностей для нестандартных решений

Слоган сервиса Bitbucket («ведро битов»):

Bitbucket is the Git solution for professional teams (Bitbucket - решение Git для профессиональных команд)

Система контроля версий Git



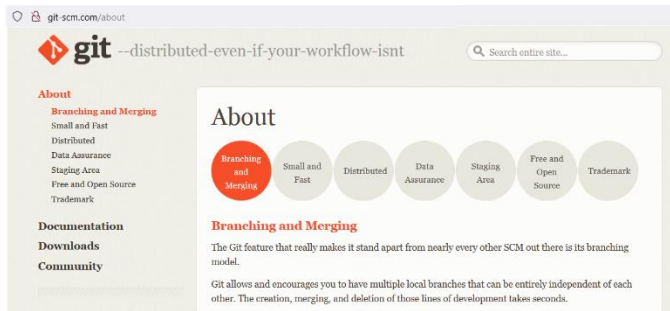
Git - распределённая система управления версиями.

Дата запуска: 2005 г.

Языки программирования:

Си, командная оболочка UNIX, Perl, Tcl, Python и C++

Разработчик: Линус Торвалдс



Основные свойства:

- ✓ быстроедействие и размер;
- ✓ безопасность и целостность (хэш SHA);
- ✓ достоверность;
- ✓ гибкость (нелинейные рабочие процессы – слияние, ветвление);
- ✓ производительность (простое ветвление);
- ✓ функциональность.

Сервис онлайн-хостинга репозитория GitHub



GitHub – сервис онлайн-хостинга репозитория, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом – всё, что поддерживает **Git**.

Дополнительно:

- ✓ обучение (глобальный поиск);
- ✓ реклама (резюме);
- ✓ контроль доступа;
- ✓ Bug трекинг;
- ✓ управление задачами;
- ✓ вики для каждого проекта.

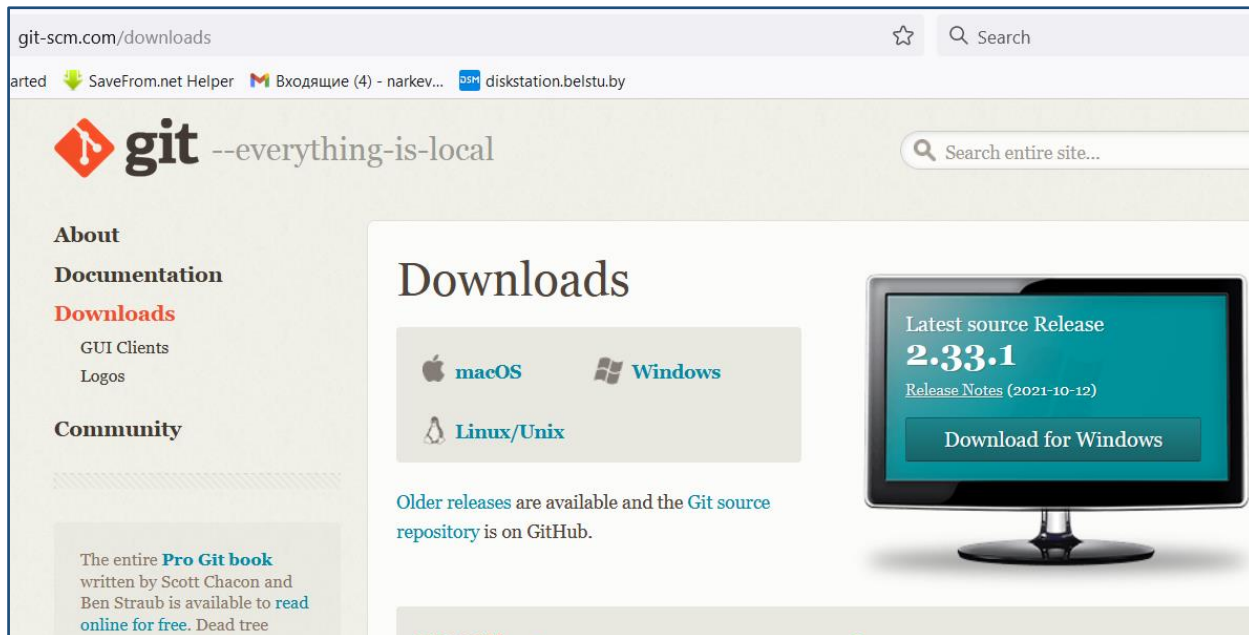
Чем отличается **Git** и **GitHub**

| | |
|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| Git: | <ul style="list-style-type: none">• инструмент, позволяющий реализовать распределённую систему контроля версий. |
| GitHub: | <ul style="list-style-type: none">• сервис для проектов, использующих Git. |

Система контроля версий Git

Установка и настройка Git

Загрузить Git можно с официального сайта: <http://git-scm.com/>



Настройка Git

Конфигурирование Git с помощью утилиты *командной строки*

Запустить Git Bash на компьютере.

Глобальными настройками являются *имя пользователя* и его *email*. Их можно установить следующими командами в консоли *Git Bash*:

```
$ git config --global user.name <"Your name">
$ git config --global user.email <email@example.com>
```

В Git существует три места, где хранятся настройки:

- на уровне системы;
- на уровне пользователя;
- на уровне проекта (репозитория).

Все параметры будут помещены в файл с настройками Git `.gitconfig`, расположенном в домашнем каталоге пользователя

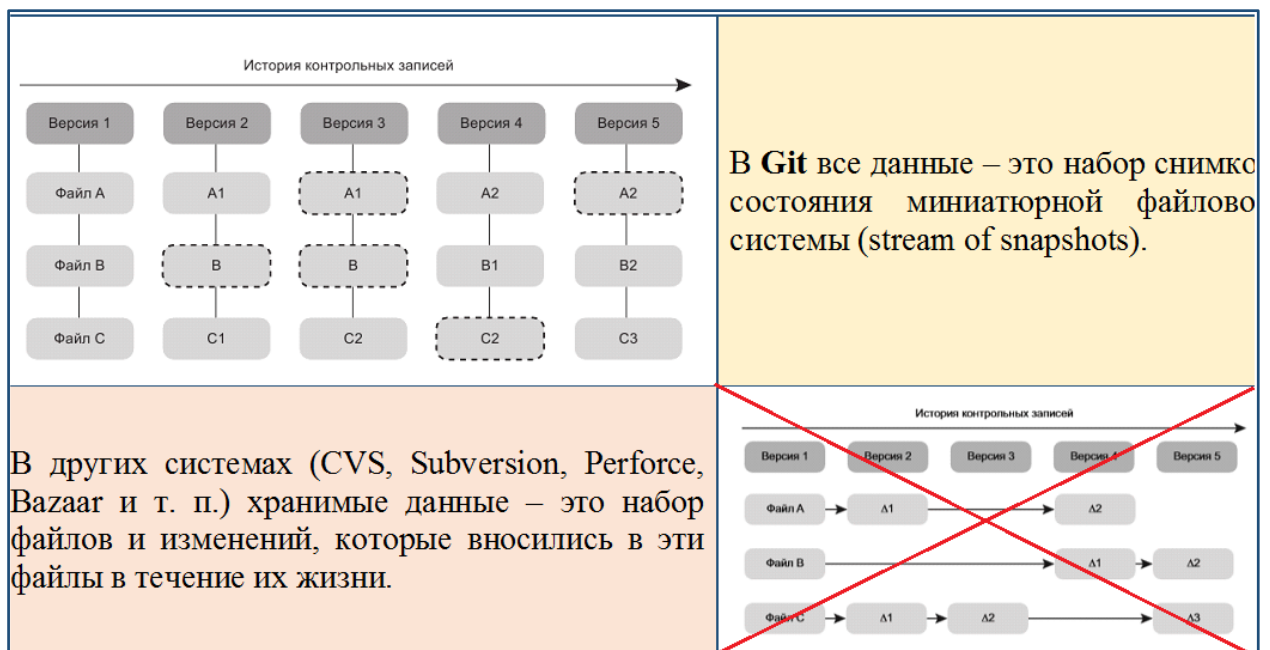
Для просмотра введенных изменений воспользуйтесь командой:

```
$ git config --list
```

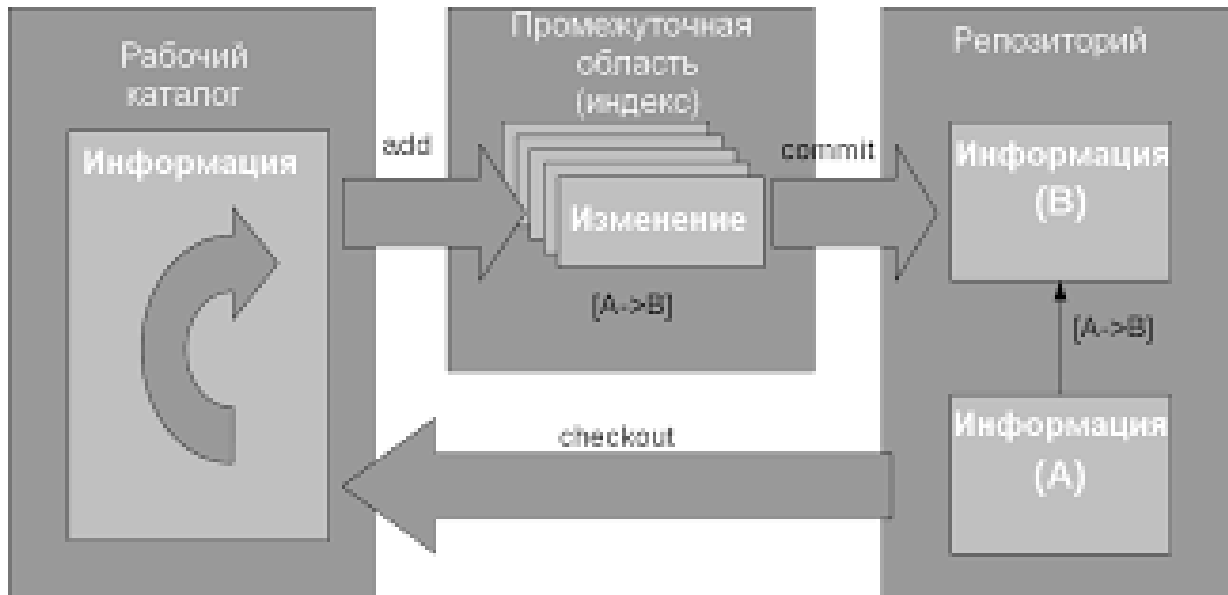
```
chimaera@W520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам
$ git config --list
pack.packsizelimit=2g
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files (x86)/Git/mingw32/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=adel
user.email=narkevich.adelina@gmail.com
```

Отличия Git от других систем контроля версий:

1) Хранит снимки состояний, а не изменений



2) Локальность операций



3) Целостность Git (вычисление контрольных сумм – хеш SHA-1)

Хеш - строка из 40 символов, включающая в себя числа в шестнадцатеричной системе (0–9 и a–f) и вычисляемая на основе содержимого файла или структуры папки в Git.

Пример:

2bcad51a4025dde7f4b7c2c28d4f4ac614964475

```
chimaera@w520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ git log
commit 2bcad51a4025dde7f4b7c2c28d4f4ac614964475 (HEAD -> master)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 3 22:45:25 2021 +0300

    added Hello.txt to the repo

chimaera@w520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ |
```

Основные определения

| | |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Репозиторий Git: | Git хранит информацию в структуре данных, называемой репозиторий (repository). Репозиторий хранится в папке проекта – в папке .git |
| Репозиторий хранит: | <ul style="list-style-type: none">– набор коммитов (commit objects)– набор ссылок на коммиты (heads). |
| Commit objects содержат: | <ul style="list-style-type: none">– набор файлов, отображающий состояние проекта в текущий момент времени– ссылки на родительские commit objects– SHA1 имя – 40 символьная строка, которая уникально идентифицирует commit object |

Основные команды

`git init` – создание репозитория

`git add` <имена файлов> – добавляет файлы в индекс

`git commit` – выполняет коммит проиндексированных файлов в репозиторий

`git status` – показывает какие файлы изменились между текущей стадией и HEAD. Файлы разделяются на 3 категории: **новые** файлы, **измененные** файлы, **добавленные** новые файлы

`git checkout` <SHA1 или метка> – получение указанной версии файла

`git push` – отправка изменений в удаленный репозиторий

`git fetch` – получение изменений из удаленного репозитория

`git clone` <remote url> – клонирование удаленного репозитория себе

Все возможные команды можно получить с помощью команды

```
$ git help
```

Фрагмент вывода справки в консоль:

```
$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        [--super-prefix=<path>] [--config-env=<name>=<envvar>]
        <command> [<args>]

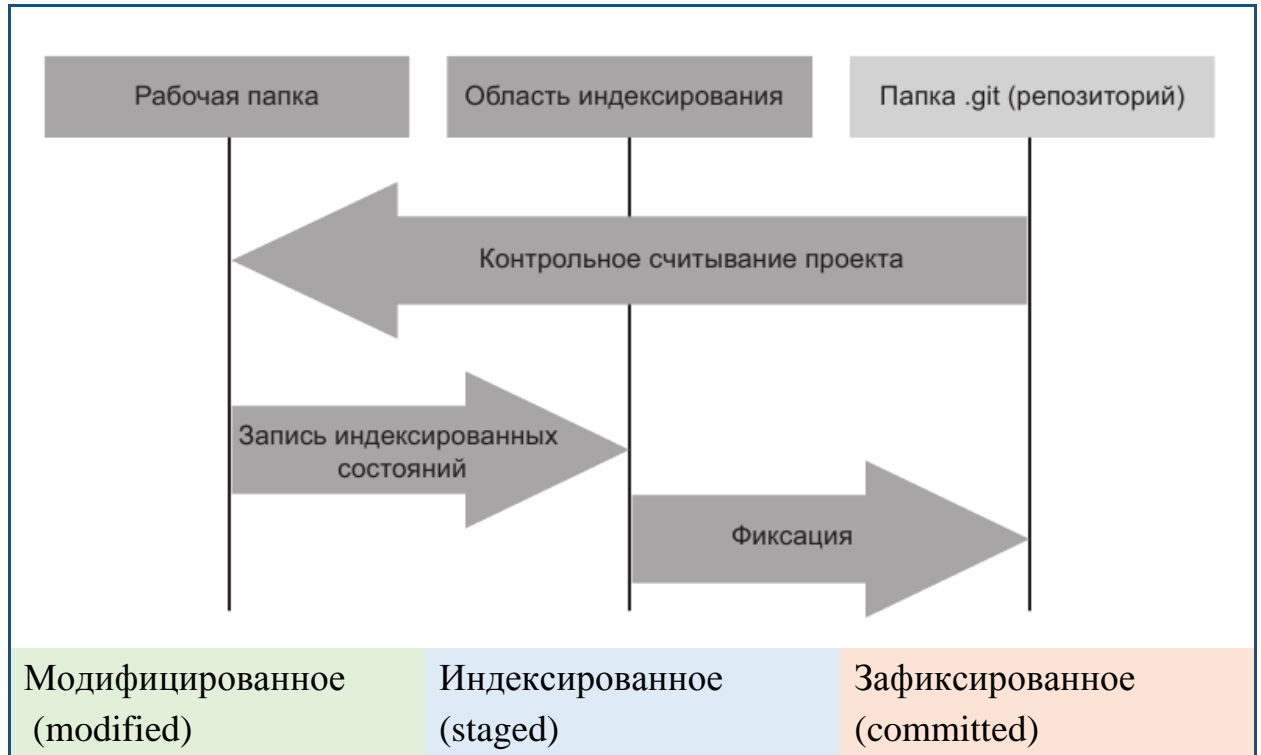
These are common Git commands used in various situations:

3 start a working area (see also: git help tutorial)
    clone          Clone a repository into a new directory
    init           Create an empty Git repository or reinitialize an existing
one

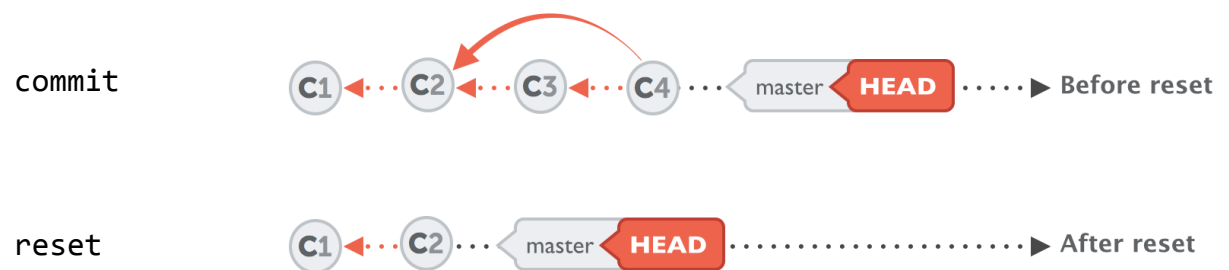
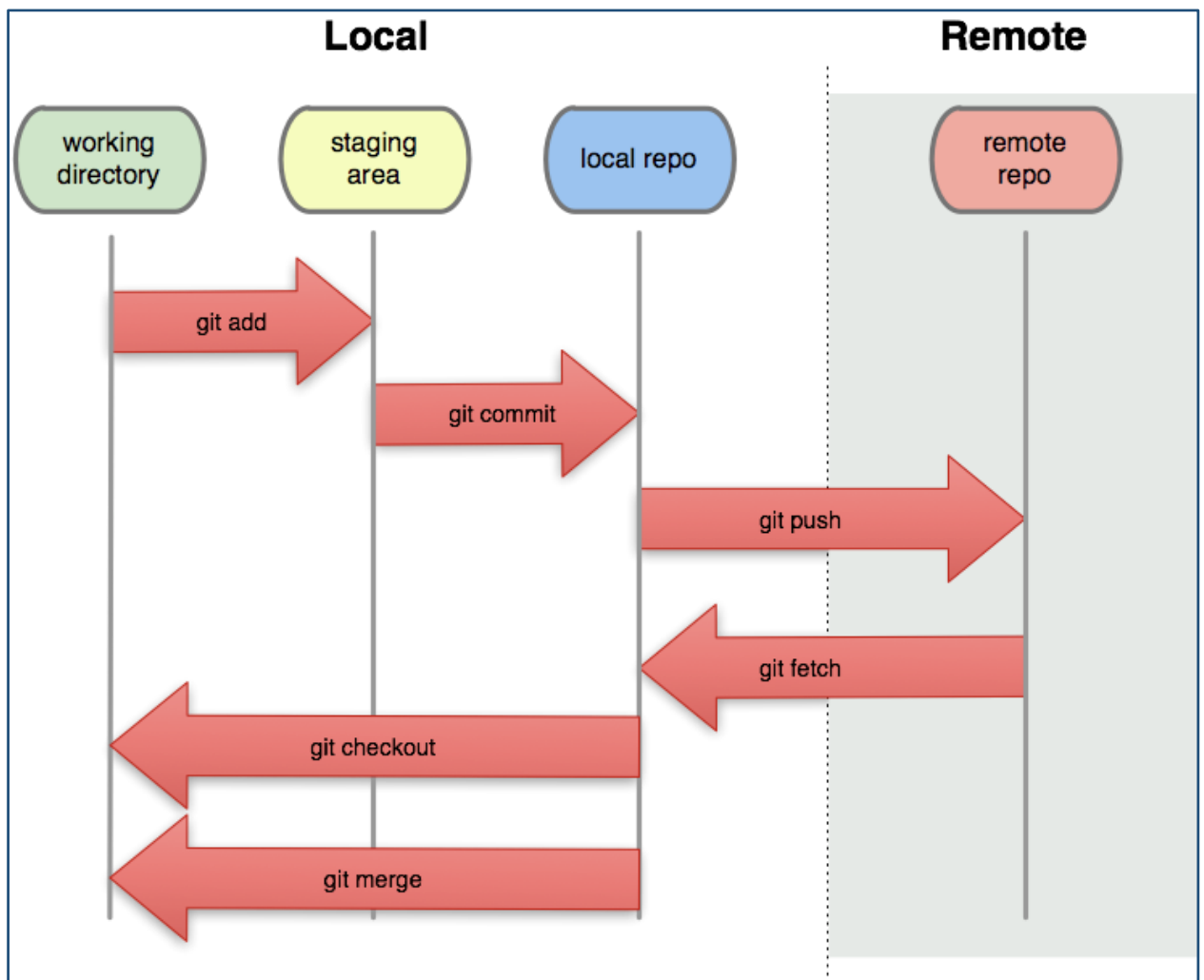
work on the current change (see also: git help everyday)
    add            Add file contents to the index
    mv             Move or rename a file, a directory, or a symlink
    restore        Restore working tree files
    rm             Remove files from the working tree and from the index
    sparse-checkout Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
```

4) Три состояния файлов



| | |
|---------------------------------------|------------------------------------------------------------------------------------|
| Модифицированное (modified) состояние | изменения уже внесены в файл, но пока не зафиксированы в базе данных |
| Индексированное (staged) состояние | текущая версия модифицированного файла помечена как требующая последующей фиксации |
| Зафиксированное (committed) состояние | данные надежно сохранены в локальной базе |



Пример создания локального репозитория:

- Перейти в проводнике в рабочую папку, где планируется создать репозиторий, и запустить Git Bash с помощью контекстного меню: «Git Bash Here».
- Инициализация репозитория в выбранной папке выполняется командой

```
$ git init
```

```
chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/К лабораторным
$ git init
Initialized empty Git repository in D:/Ade1/Кафедра/ОПИ+ТРПО/К лабораторным/.git/
chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/К лабораторным (master)
$ |
```

В папке появилась новая скрытая папка: **.git** – локальный репозиторий

| ОПИ+ТРПО ▸ Примеры к лабораторным работам | | | |
|-------------------------------------------|------------------|--------------------|--------|
| Имя | Дата изменения | Тип | Размер |
| .git | 03.11.2021 22:45 | Папка с файлами | |
| HelloWorld | 13.10.2021 20:41 | Папка с файлами | |
| Lec08 | 27.10.2021 23:50 | Папка с файлами | |
| Hello.txt | 03.11.2021 22:32 | Текстовый докум... | 1 КБ |

Текущее состояние (**status**) репозитория отображается командой:

```
$ git status
```

Кроме того, была создана ветка **master**:

```
chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ git status
On branch master

No commits yet

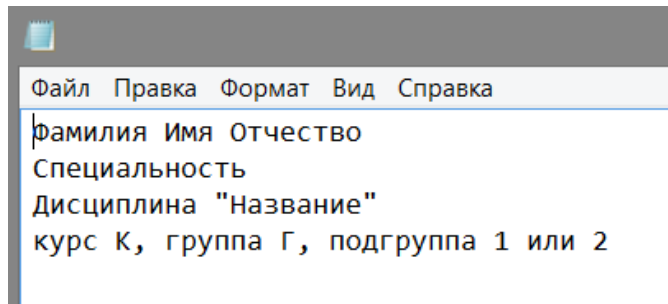
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  Hello.txt
  HelloWorld/
  Lec08/

nothing added to commit but untracked files present (use "git add" to track)
```

Красным цветом отмечены новые и модифицированные файлы и папки.

Сохранение изменений в репозитории

В папке находится файл `Hello.txt` со следующим содержимым:

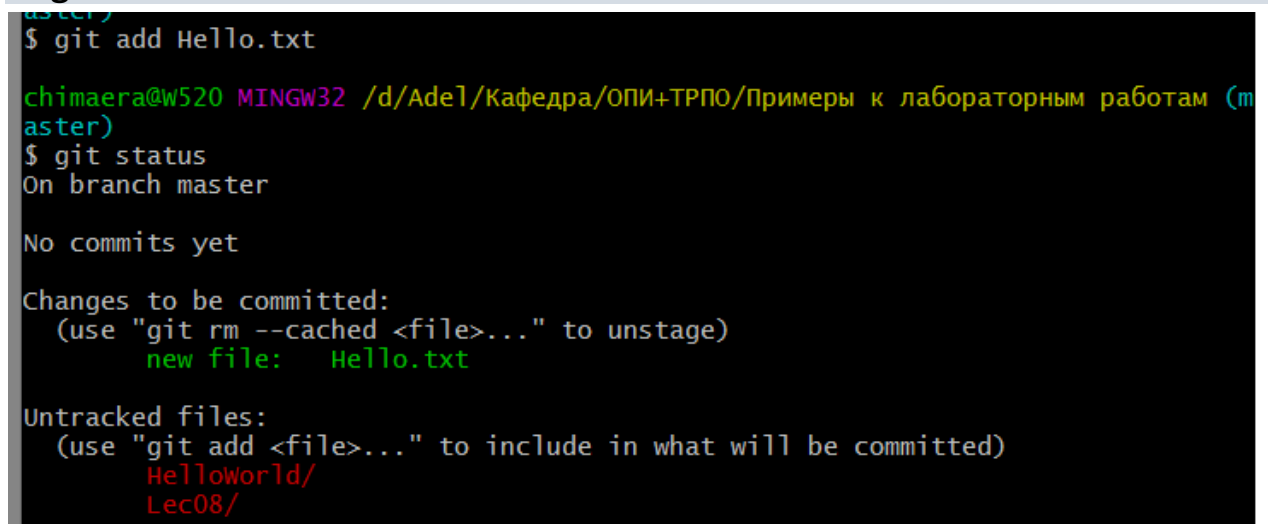


Добавим файл `Hello.txt` в репозиторий индексированных файлов командой:

```
$ git add Hello.txt
```

Текущее (обновленное) состояние репозитория отображается командой:

```
$ git status
```

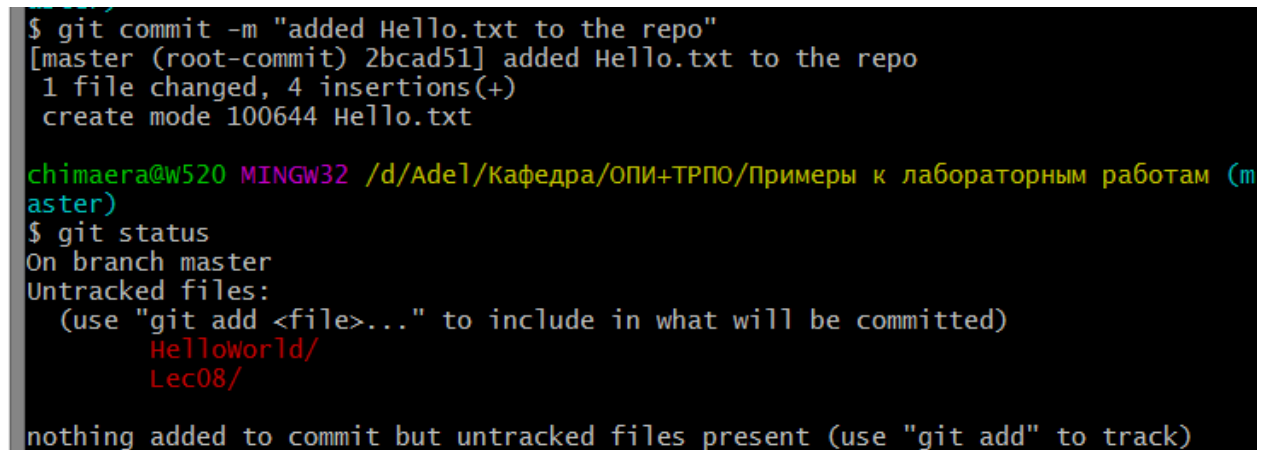


Теперь файл `Hello.txt` проиндексирован.

Эти изменения можно зафиксировать в репозитории командой:

```
$ git commit -m "added Hello.txt to the repo"
```

Ключ `-m` позволяет добавить комментарий, описывающий, что именно было изменено в коммите ("added Hello.txt to the repo")



Git информирует об успешном создании нового коммита, в ветку master добавлен 1 файл:

```
[master (root-commit) 2bcad51] added Hello.txt to the repo  
1 file changed, 4 insertions(+)
```

Теперь состояние файла `Hello.txt` зафиксировано в репозитории.

Все коммиты в Git логируются. Просмотреть журнал можно с помощью команды

```
$ git log
```

которая показывает лог `commits` начиная с указателя HEAD

```
$ git log  
commit 2bcad51a4025dde7f4b7c2c28d4f4ac614964475 (HEAD -> master)  
Author: adel <narkevich.adelina@gmail.com>  
Date:   Wed Nov 3 22:45:25 2021 +0300  
  
    added Hello.txt to the repo
```

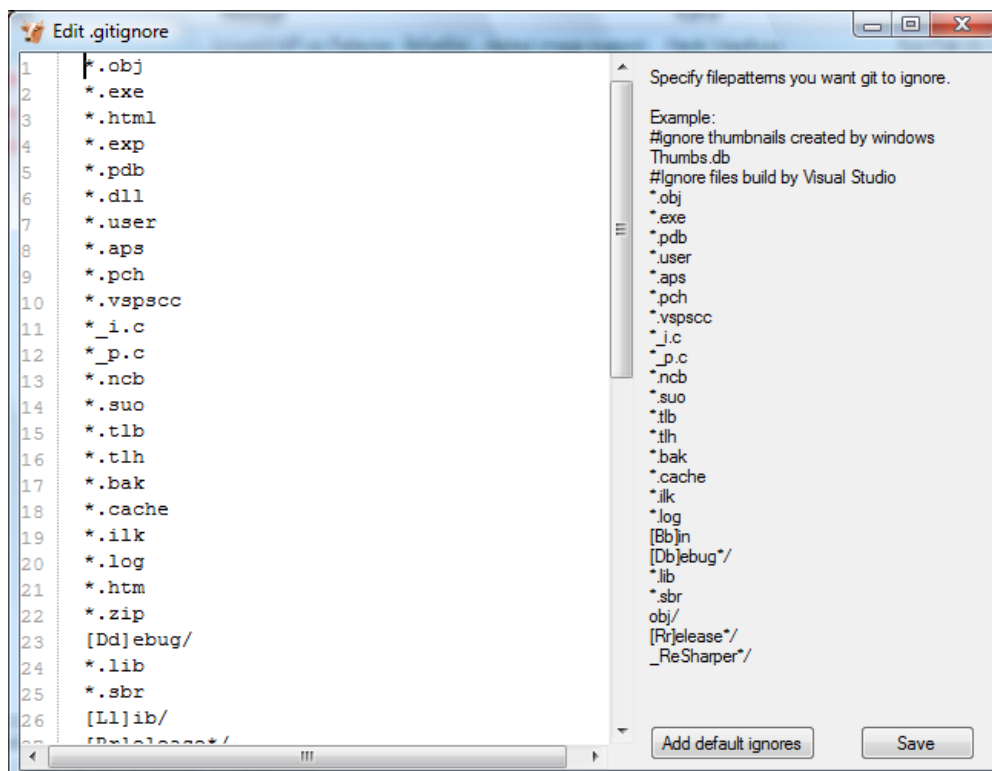
Игнорирование файлов

Не все файлы проекта требуется включать в систему контроля версий. Можно исключить:

- настройки IDE;
- результаты сборки;
- файлы кэша;
- индивидуальные файлы пользователя и др.

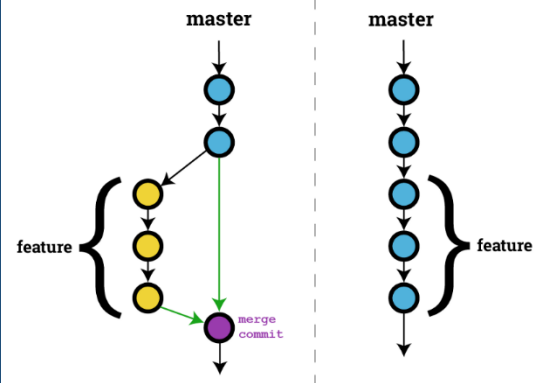
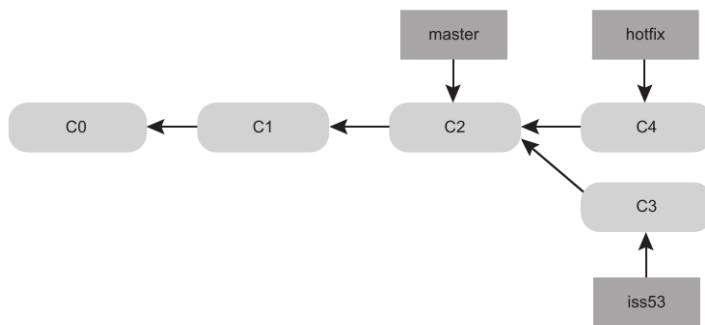
Ознакомиться с шаблонами `.gitignore` (текстовый файл) можно по ссылке:

<https://github.com/github/gitignore>

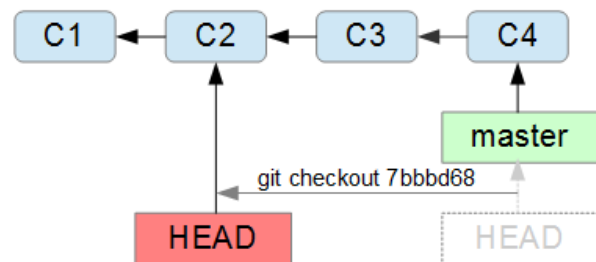
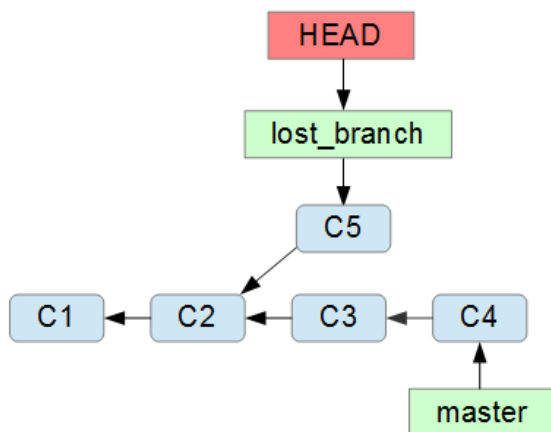


5) Ветвления

Ветвление (branching) означает отклонение от основной линии разработки, после которого работа перестает затрагивать основную линию и переходит в ветвь.



Понятие HEAD



Смена веток меняет файлы в рабочей папке

Работа с ветками

Создание ветки (branch) выполняется следующей командой:

```
$ git branch <branch_name>
```

```
$ git brabch
git: 'brabch' is not a git command. See 'git --help'.

The most similar command is
    branch
```

Просмотреть список всех веток и определить текущую можно командой:

```
$ git branch
```

```
$ git branch
* master
  test
```

Переключение веток осуществляется командой:

```
$ git checkout <branch_name>
```

```
$ git checkout test
Switched to branch 'test'

chimaera@W520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (test)
$ git branch
  master
* test
```

В результате выполнения команды указатель HEAD сдвигается на ветку test.

Команда переключения веток выполняет 2 функции:

- ✓ сдвигает указатель HEAD на branch_name
- ✓ перезаписывает все файлы в папке на соответствующие новому HEAD

```
$ git status
On branch test
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    HelloWorld/
    Lec08/
    Test.txt

nothing added to commit but untracked files present (use "git add" to track)
```

HEAD указывает на ветку test:

```
$ git log
commit 2bcad51a4025dde7f4b7c2c28d4f4ac614964475 (HEAD -> test, master)
Author: adel <narkevich.adelina@gmail.com>
Date:   Wed Nov 3 22:45:25 2021 +0300

    added Hello.txt to the repo
```

Индексируем новый файл и фиксируем изменения в репозитории.

```
$ git status
On branch test
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Test.txt

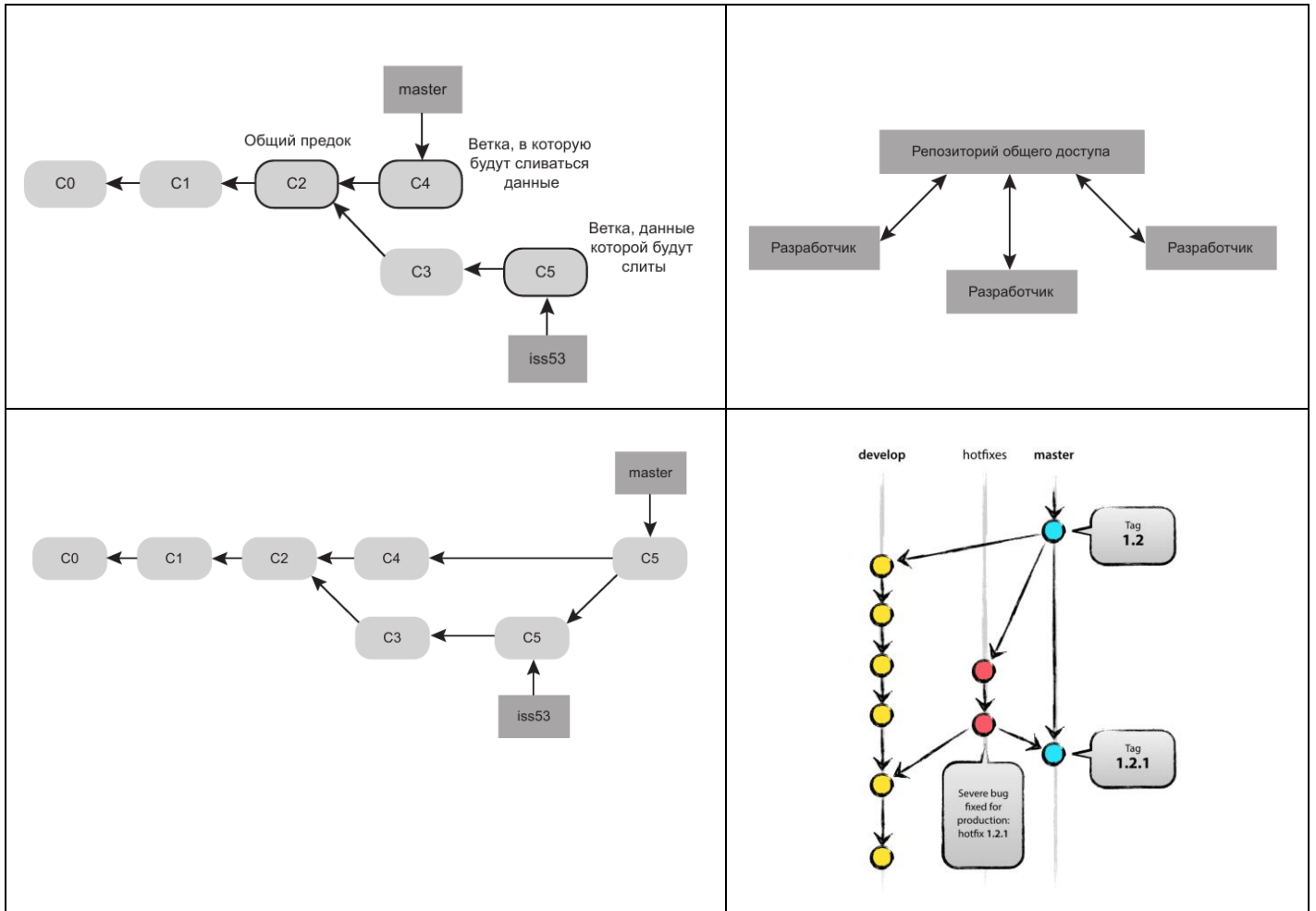
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    HelloWorld/
    Lec08/

chimaera@w520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (test)
$ git commit -m "added file Test.txt"
[test 621e67b] added file Test.txt
1 file changed, 1 insertion(+)
create mode 100644 Test.txt

chimaera@w520 MINGW32 /d/Adel/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (test)
$ git status
On branch test
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    HelloWorld/
    Lec08/

nothing added to commit but untracked files present (use "git add" to track)
```

6) Слияния веток



Переходим в ветку, в которой будет выполняться слияние (master)

```
chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (test)
$ git checkout master
Switched to branch 'master'

chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ git merge test
Updating 2bcad51..621e67b
Fast-forward
 Test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Test.txt

chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        HelloWorld/
        Lec08/

nothing added to commit but untracked files present (use "git add" to track)

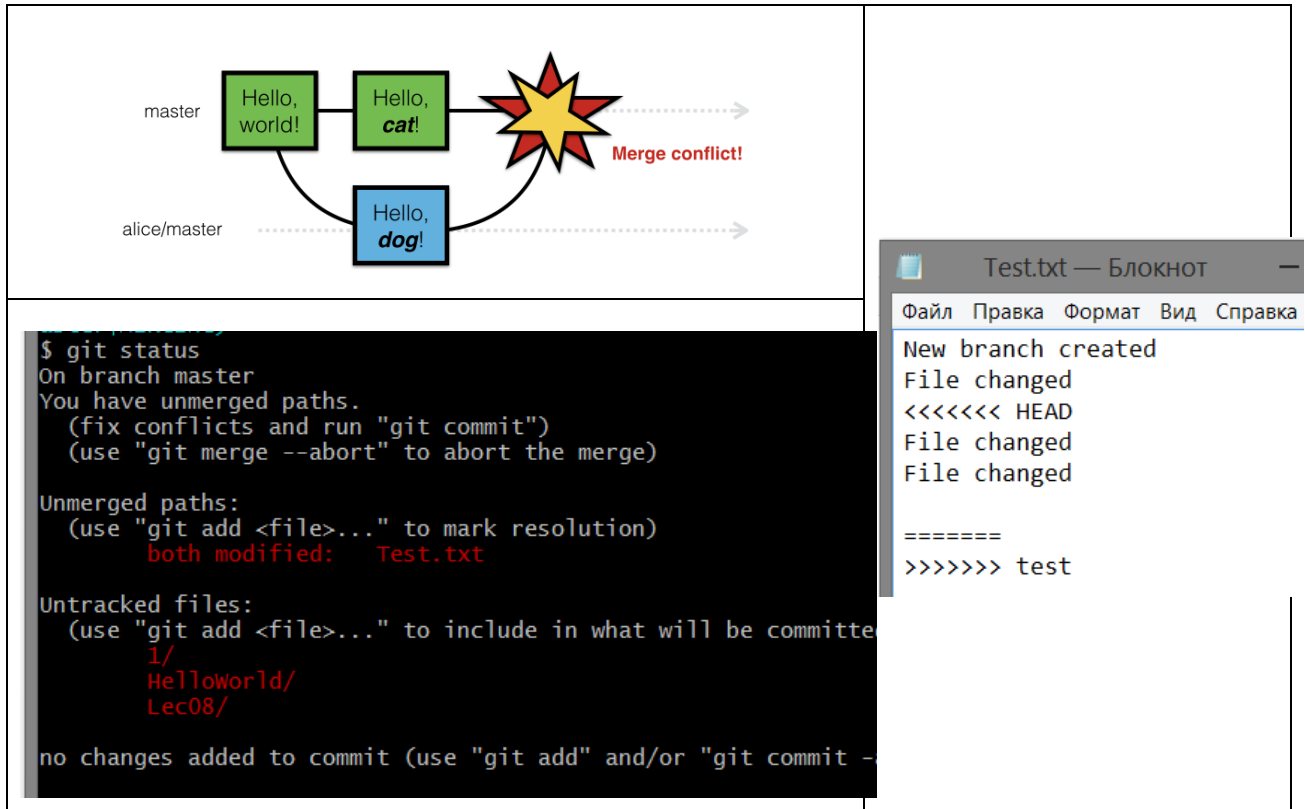
chimaera@W520 MINGW32 /d/Ade1/Кафедра/ОПИ+ТРПО/Примеры к лабораторным работам (master)
$ git log
commit 621e67b63da53aa795ae89e1730687252d71ca50 (HEAD -> master, test)
Author: adel <narkevich.adelina@gmail.com>
Date: Thu Nov 4 02:09:56 2021 +0300

    added file Test.txt

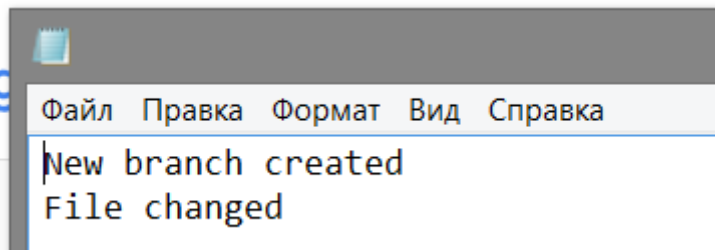
commit 2bcad51a4025dde7f4b7c2c28d4f4ac614964475
Author: adel <narkevich.adelina@gmail.com>
Date: Wed Nov 3 22:45:25 2021 +0300

    added Hello.txt to the repo
```

Конфликты при слиянии



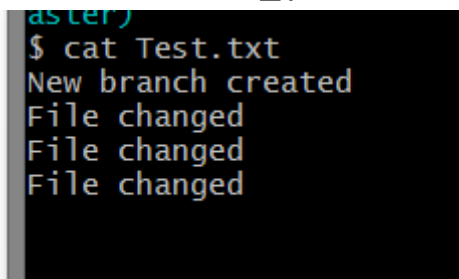
а) Файл Test.txt изменен в ветке test:



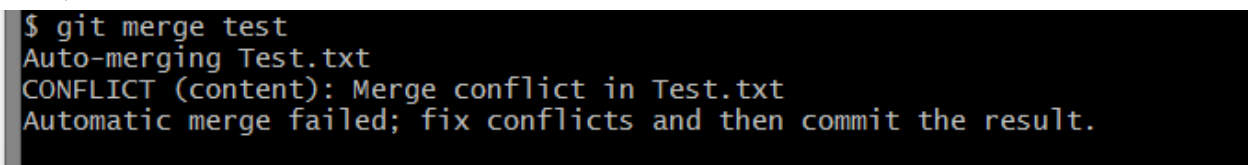
б) Изменения зафиксированы в репозитории.

с) Перешли в ветку Master и изменили файл Test.txt

д) Присмотр содержимого файла в ветке master с помощью команды `cat <имя_файла>`:



е) Пытаемся выполнить слияние веток



Git информирует о конфликте слияния веток. Git не создал коммит слияния автоматически. Процесс остановлен до тех пор, пока вы не разрешите конфликт.

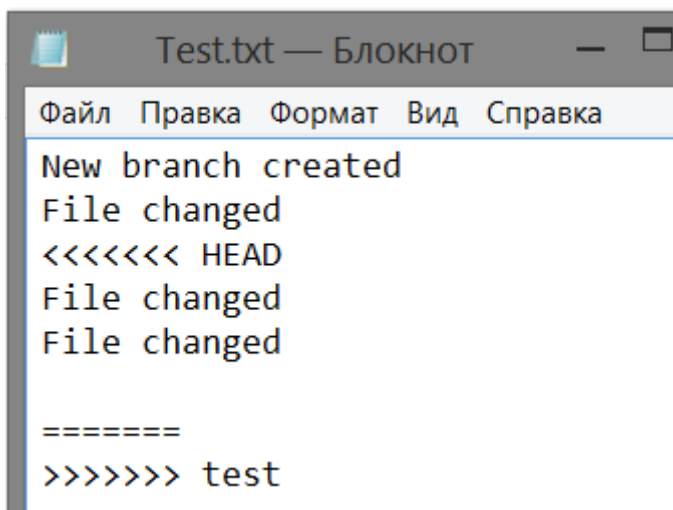
```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   Test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    1/
    Helloworld/
    Lec08/

no changes added to commit (use "git add" and/or "git commit -a")
```

f) Просмотр файла в папке репозитория:



Показаны различия текста в ветке test и master (помечены несовпадающие места в файле для двух веток:

Начало: <<<<<< HEAD

Конец: =====).

Конфликт разрешается разработчиком вручную.