Дорожная карта

0. Приступаем
1. Введение в Git
2. Начало работы с Git
3. Просмотр истории
4. Ветвление
5. Слияние 👈
6. Отмена изменений
7. Рабочий процесс

Дорожная карта (продолжение)

8. Работа в команде
9. Метки
10. Последние штрихи
11. Завершаем

Слияние

Слияние - это операция, когда изменения *из другой* ветки вливаются *в текущую* ветку.

Новый коммит создаётся в текущей ветке. Ветка, из которой брали изменения, не изменяется и остаётся как есть.

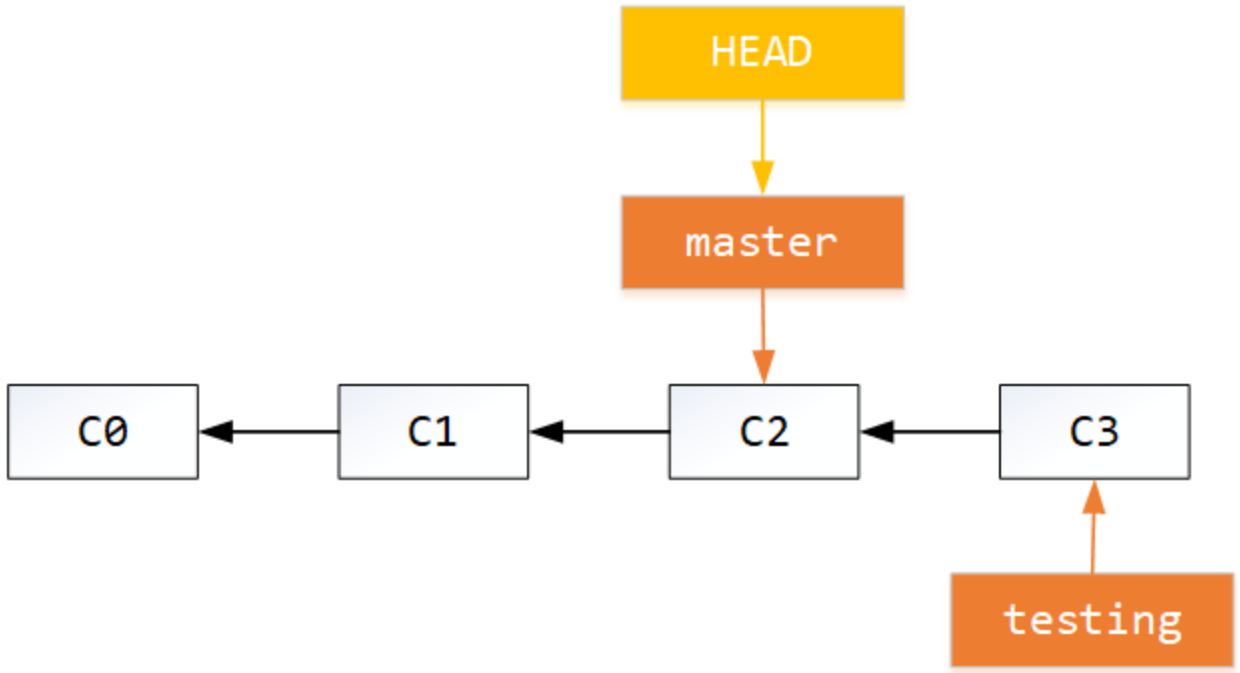
Есть два вида слияния:

- быстрая перемотка (fast-forward)
- через отдельный коммит

Быстрая перемотка (fast-forward)

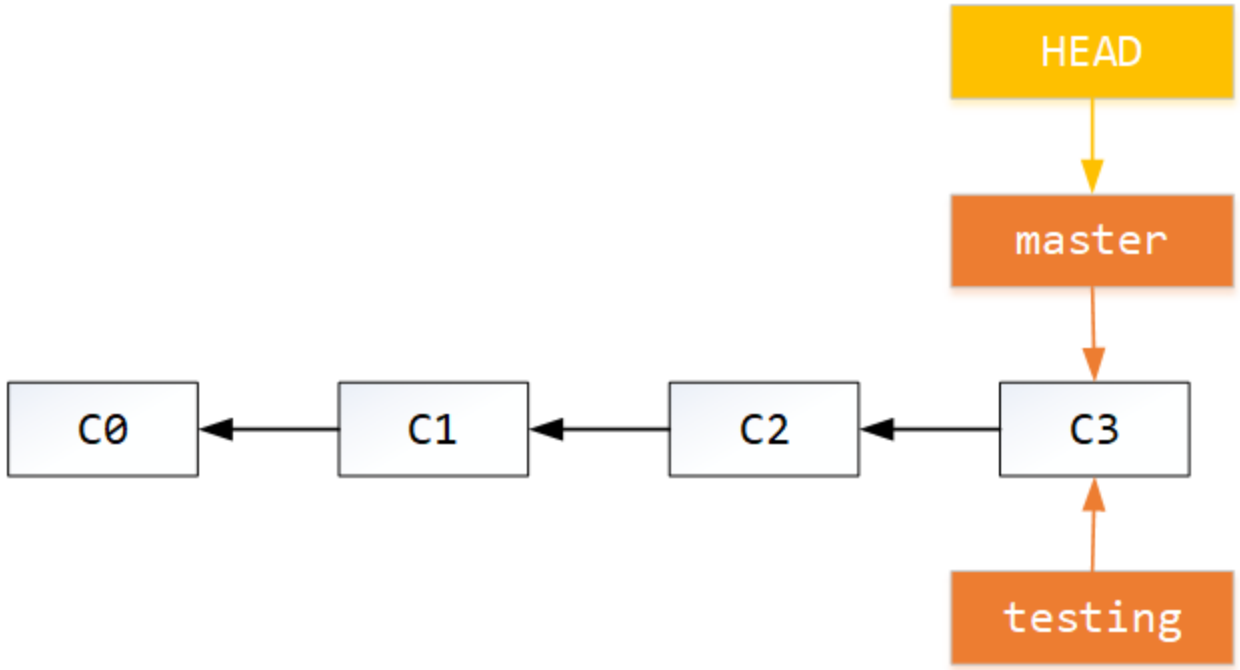
Используется, если вливаевая ветка находится дальше по истории *без* ветвлений. Указатель текущей ветки перемещается на указатель вливаемой ветки.

До:



Быстрая перемотка (fast-forward) (продолжение)

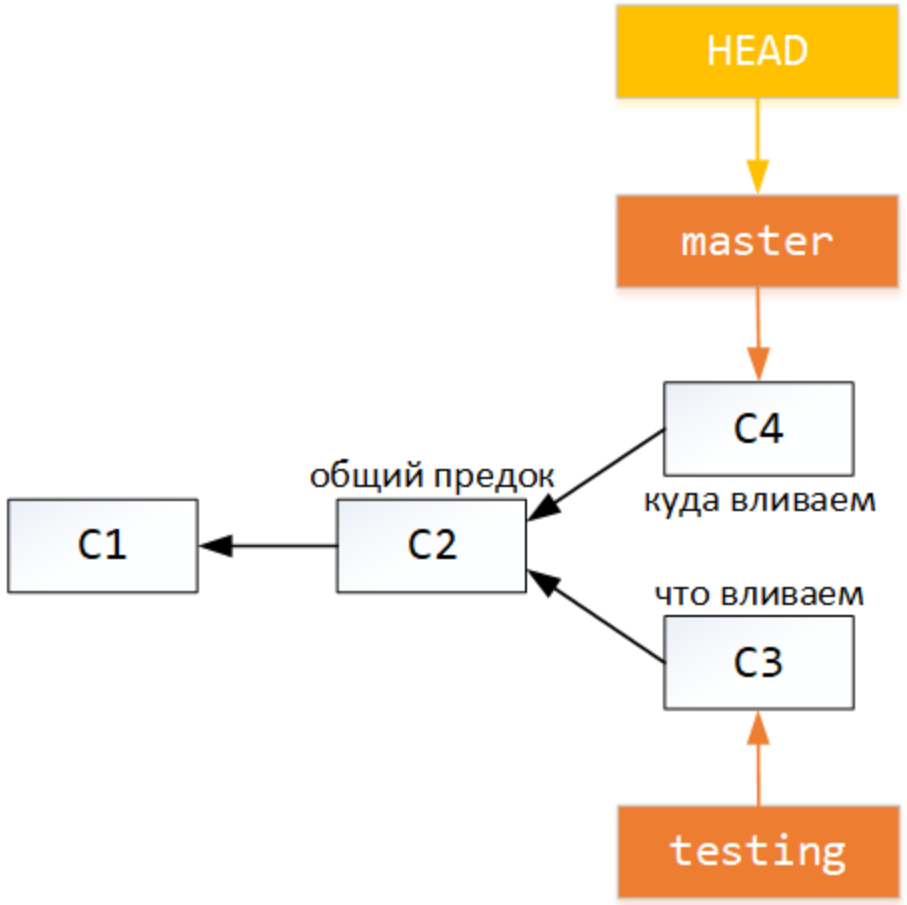
После:



Через отдельный коммит

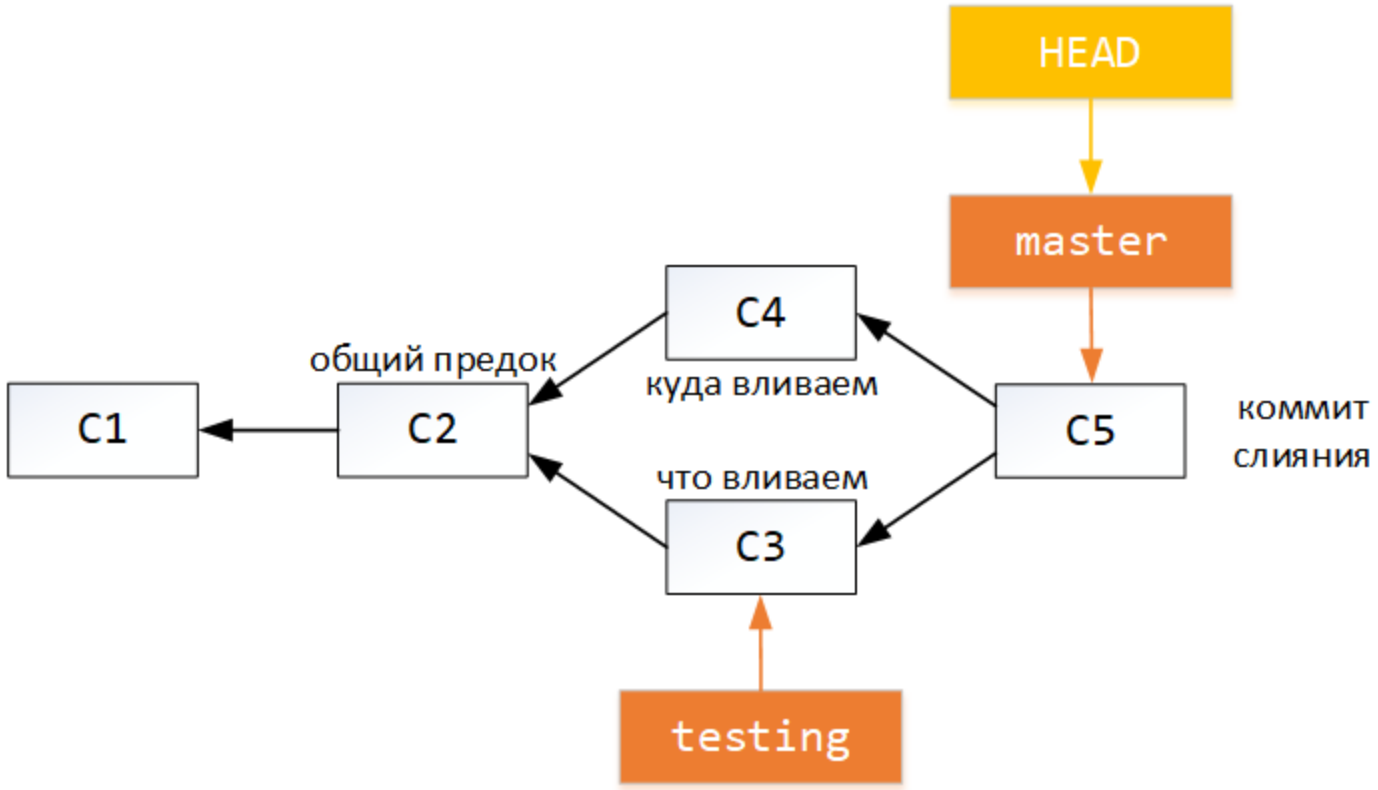
Используется, если история веток разошлась - в каждой ветке есть свои коммиты.
При слиянии создаётся новый коммит, который указывает на коммиты из веток.

До:



Через отдельный коммит (продолжение)

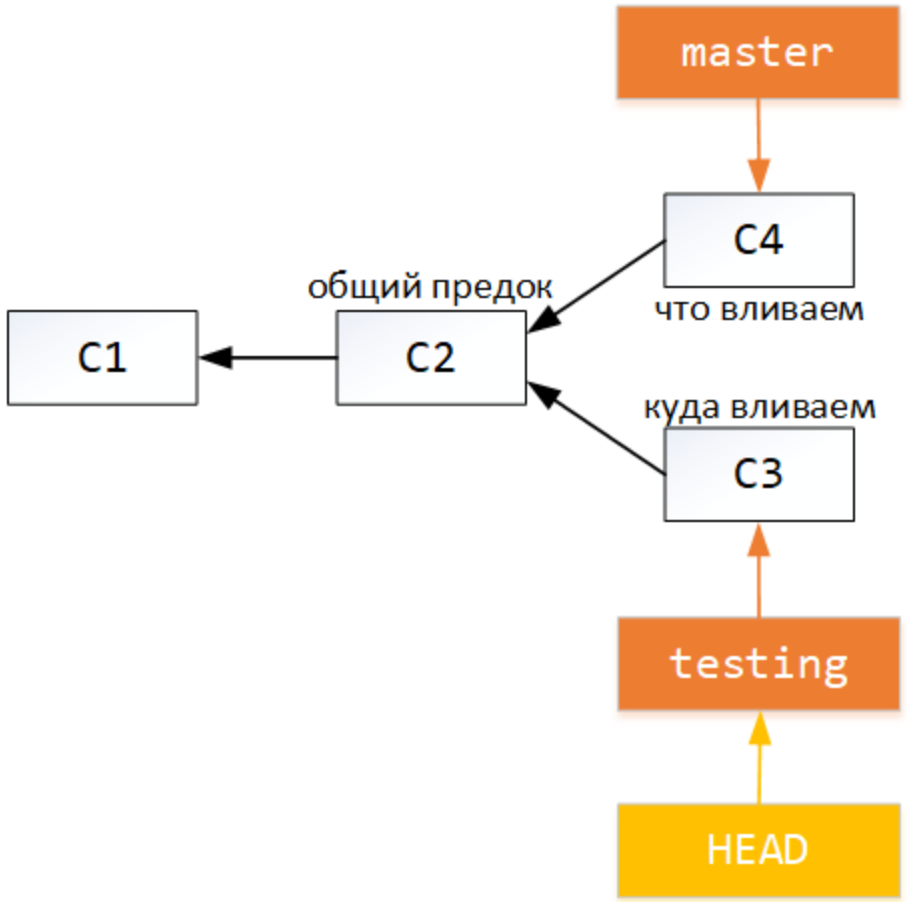
После:



Через отдельный коммит (продолжение)

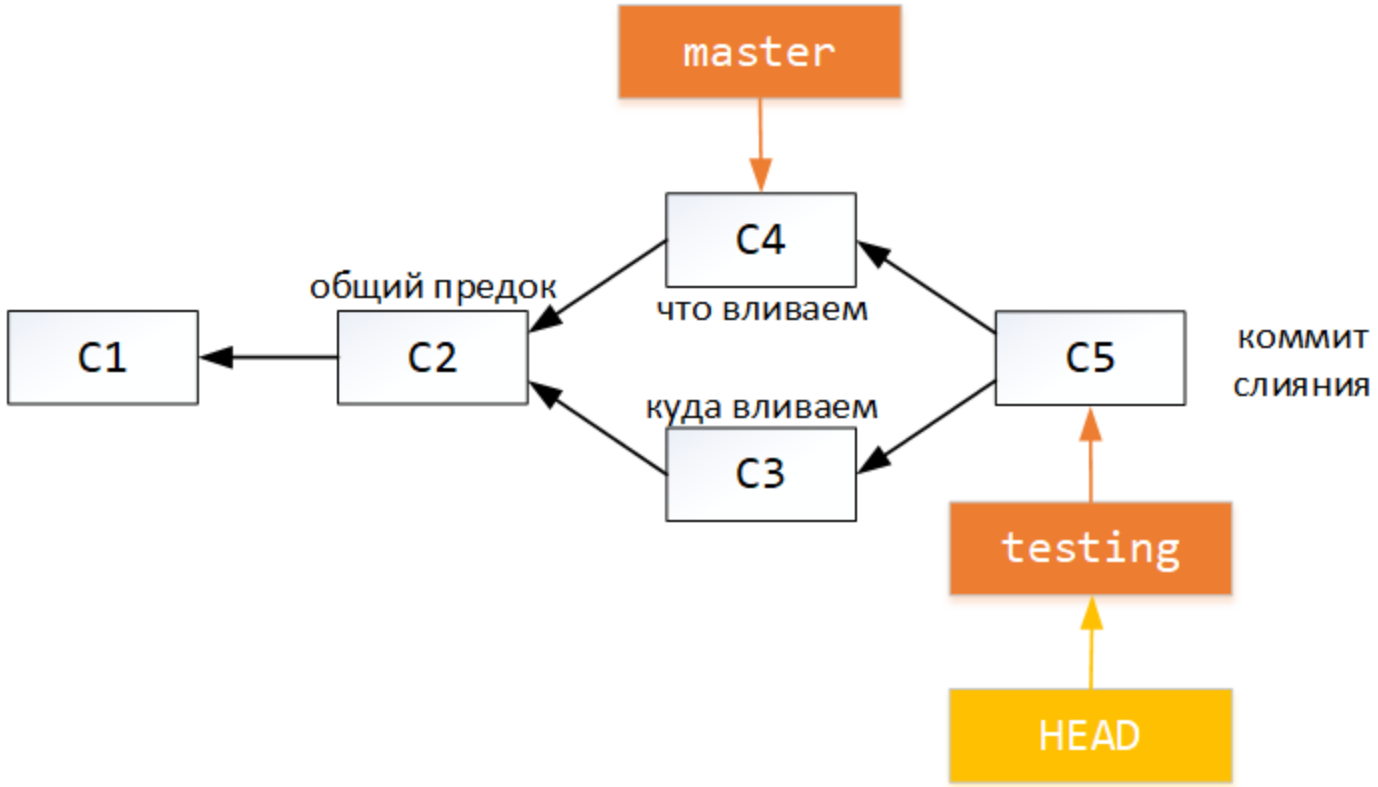
Важно выбирать, какая ветка является *текущей*.

До:

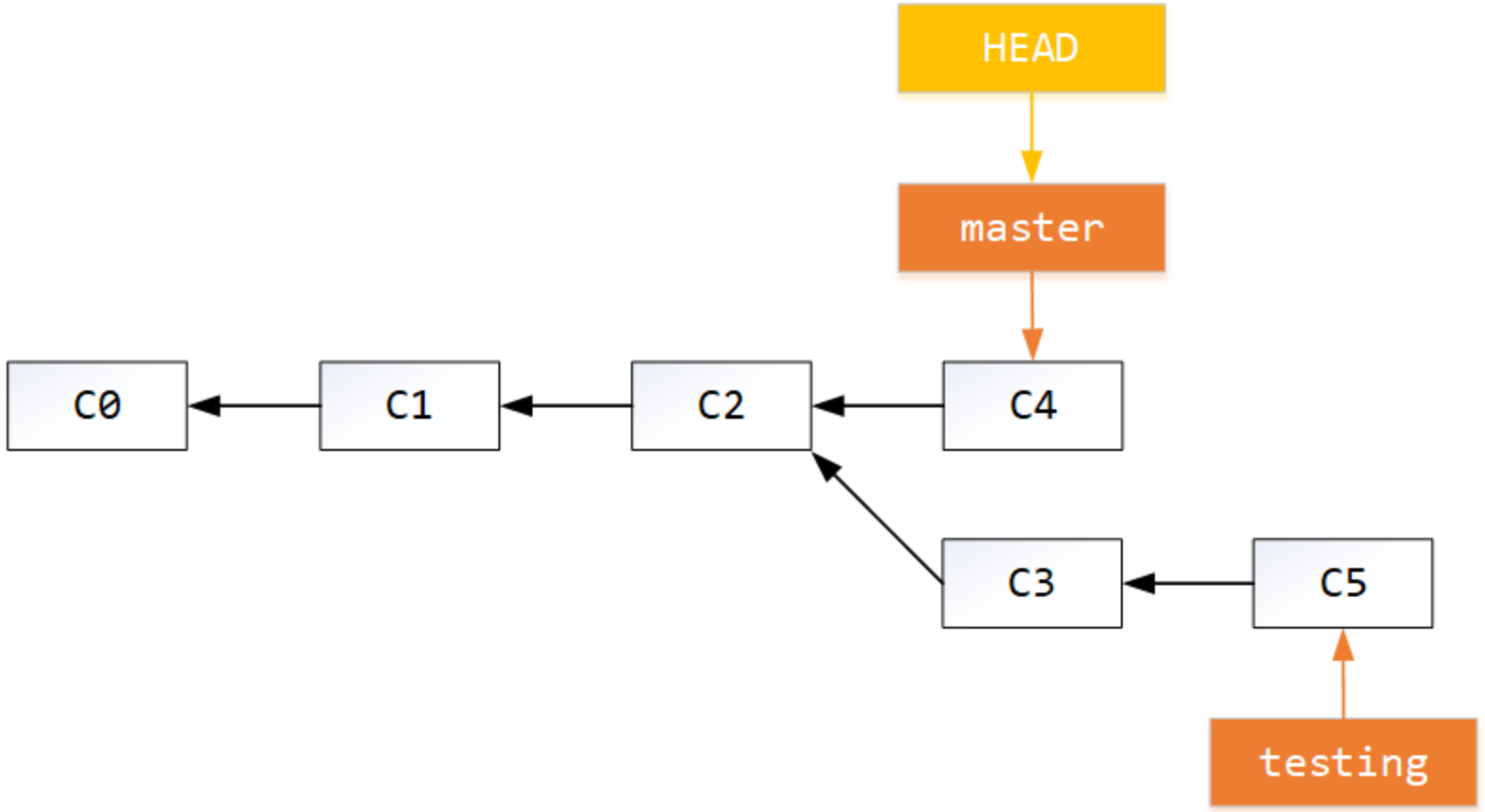


Через отдельный коммит (продолжение)

После:

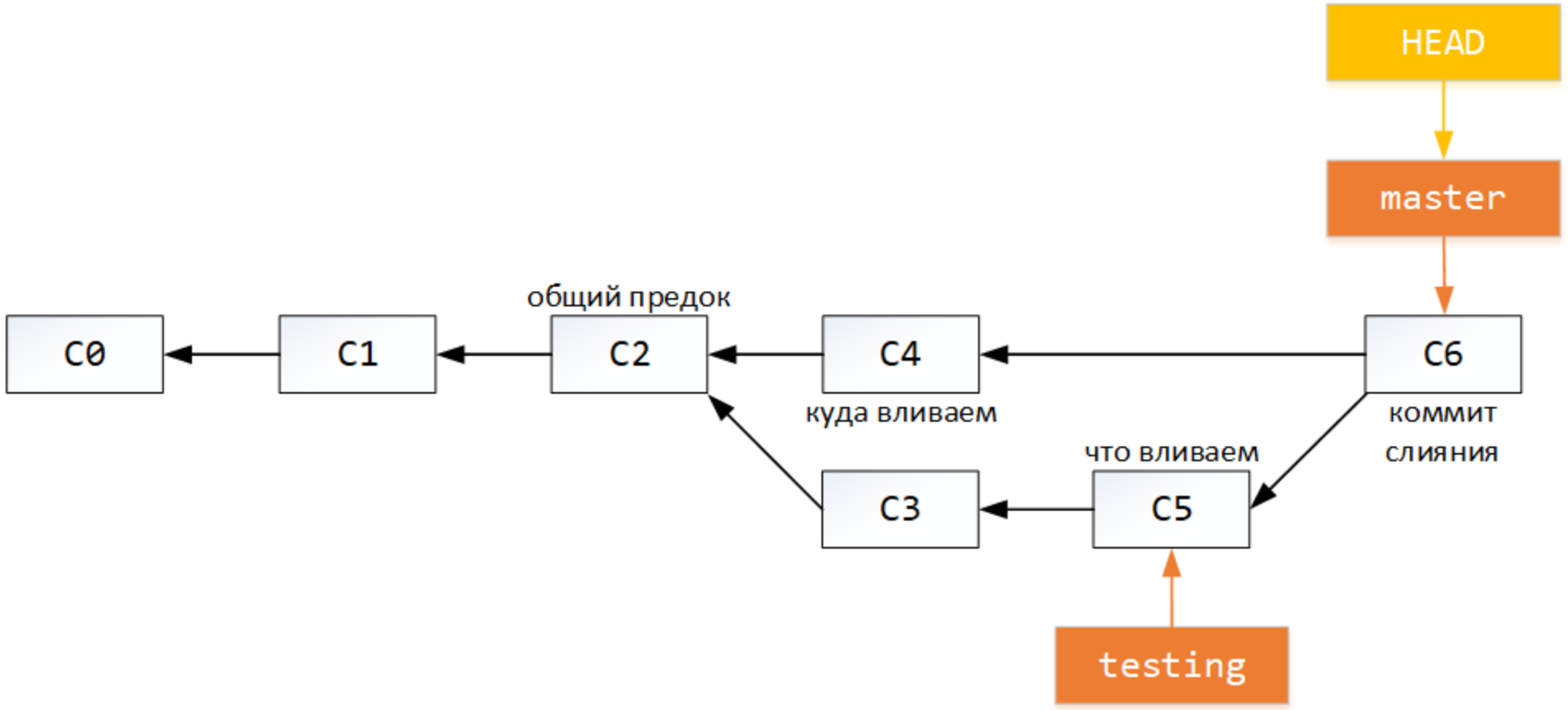


Пример → Перед слиянием



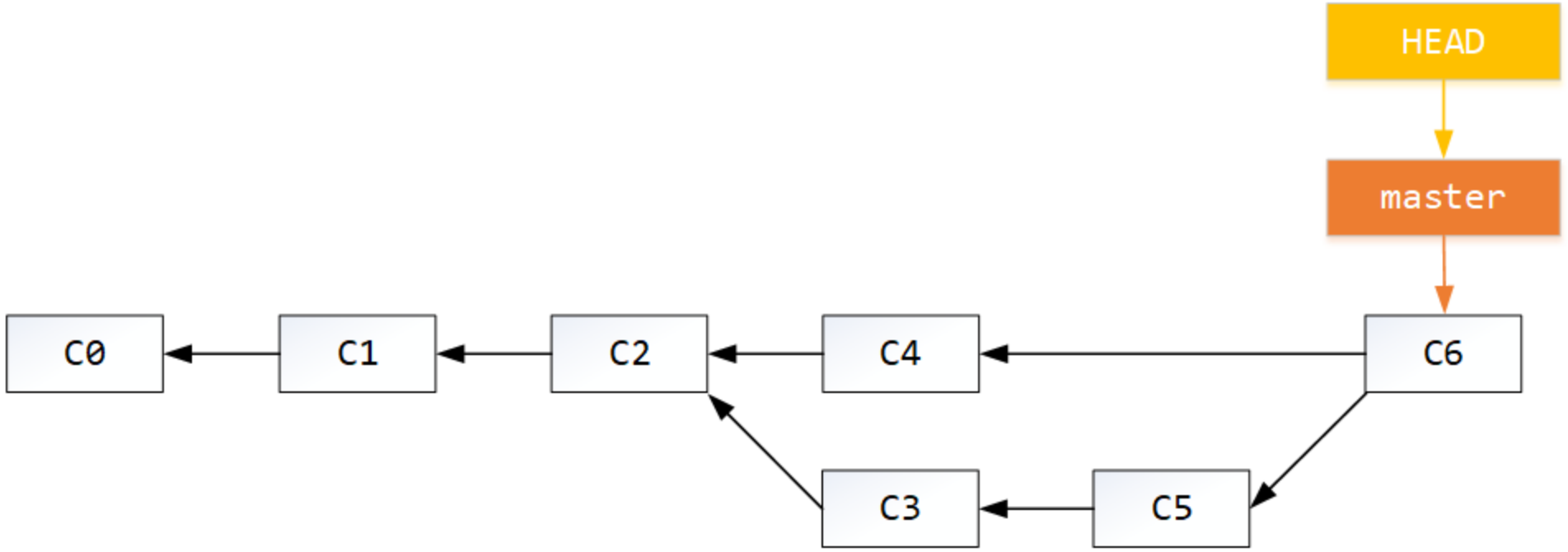
Пример → Само слияние

```
$ git merge testing
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```



Пример → Удаление ветки testing

```
$ git branch -d testing
```



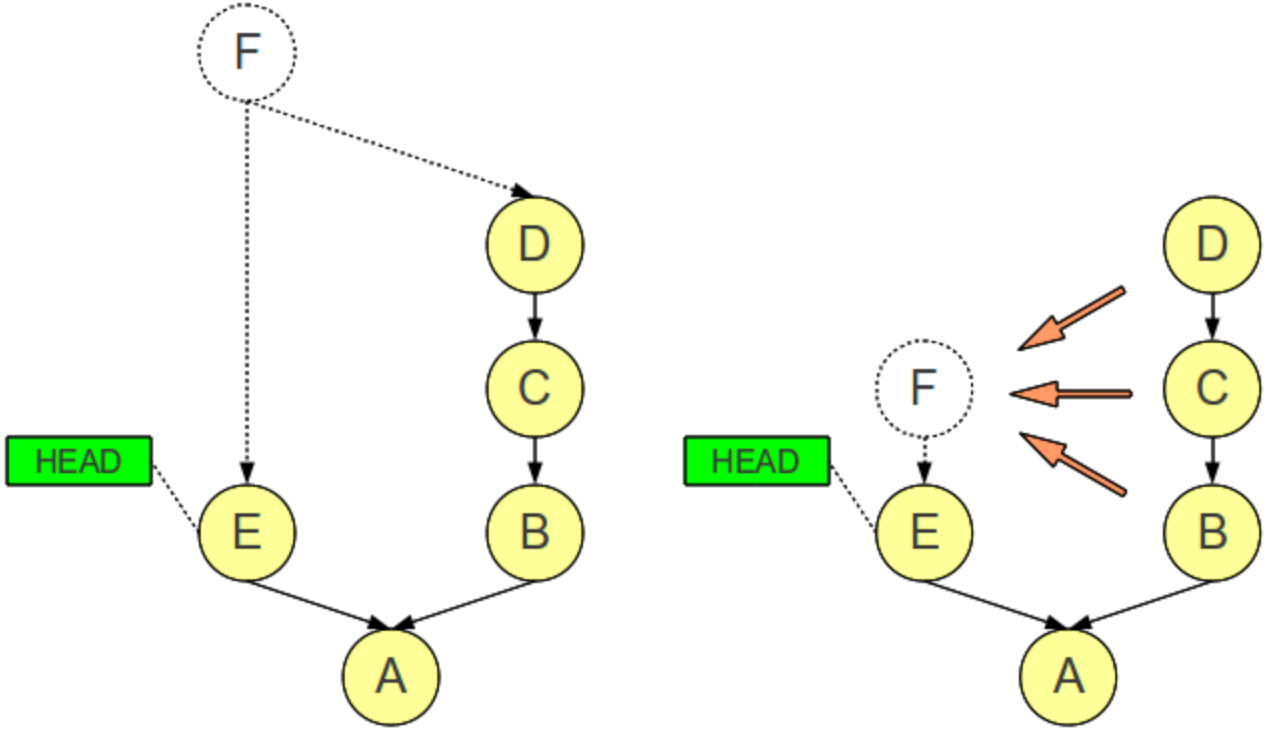
Практика 💪

1. Сделать ветку `hotfix` на вершине ветки `master`
2. Добавить несколько коммитов в `hotfix`
3. Переключиться обратно на ветку `master`
4. Сделать слияние fast-forwarding веток `master` и `hotfix`
5. Удалить ветку `hotfix`

🤔 Почему сразу нельзя делать изменения на `master`?

Слияние со сжатием коммитов

```
$ git merge --squash another-branch
$ git commit -m "<your message>"
```



Практика 💪

1. Сделайте ветку `hotfix` на предыдущем коммите `master`

```
$ git checkout -b hotfix master~1
```

2. Сделайте несколько коммитов на ветке `hotfix`, добавляя в каждом по одному файлу.
3. Влейте изменения в ветке `master`, используя `git merge --squash`.
4. Создайте коммит на ветке `master`, чтобы зафиксировать изменения.

Долгоживущие ветки

Долгоживущие ветки - ветки, в которых разработка может идти неделями или даже месяцами.

Их регулярно нужно обновлять с основной веткой разработки (обычно `master` или `develop`). Иначе они могут сильно разойтись по изменениям и влить их будет практически невозможно.

Долгоживущие ветки → Подкачка из основной ветки

1. Делаются изменения в `myfeature`
2. Кто дописывает коммиты в `master`
3. Периодически коммиты из `master` вливаются в `myfeature`

Шаги 1-3 выполняюся на протяжении длительного времени.

4. Происходит финализация изменений в `myfeature`
5. Происходит слияние `myfeature` → `master`
6. Ветка `myfeature` удаляется

Долгоживущие ветки → Взаимное слияние

1. сперва основная ветка разработки вливается в долгоживующую (например, `master` → `myfeature`)
2. потом происходит тестирование
3. затем вливается долгоживующая в основную ветку разработки (например, `myfeature` → `master`)

Шаги 1-3 выполняются на протяжении длительного времени.

🤔 А что если кто-то успеет "накоммитить" в `master` во время тестирования?

4. Происходит финализация изменений в `myfeature`
5. Происходит слияние `myfeature` → `master`
6. Ветка `myfeature` удаляется

Конфликт слияния

Конфликт слияния - это ситуация, когда одна и та же строчка была изменена в двух ветках.

```
$ git merge testing
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

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

    both modified:      index.html

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

Разрешение конфликта

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> testing:index.html
```

```
<div id="footer">
please contact us at email.support@github.com
</div>
```

Прерывание слияния

Обычная попытка отменить слияние:

```
$ git merge --abort
```

Когда ничего не помогает:

```
$ git reset --hard HEAD
```

Отмена слияния

Если вы уже выполнили коммит после слияния, и вы хотите сбросить его:

```
$ git reset --hard HEAD~1
```

Графический инструмент разрешения конфликта

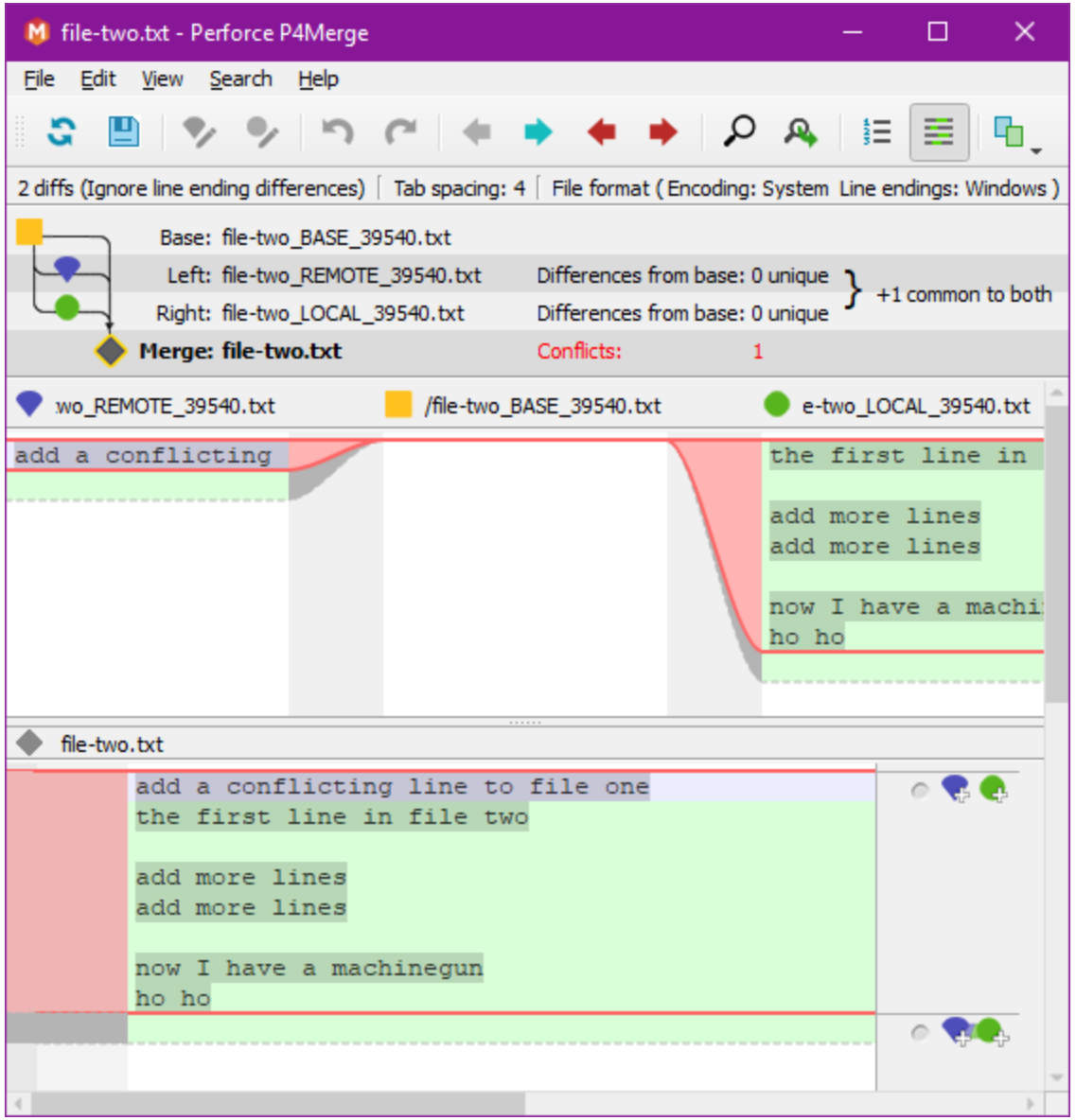
```
$ git mergetool --tool-help
'git mergetool --tool=<tool>' may be set to one of the following:
    meld
    p4merge
    vimdiff
    vimdiff1
    vimdiff2
    vimdiff3

  user-defined:
    bc.cmd "C:/Program Files/Beyond Compare 4/bcomp.exe" "$LOCAL" "$REMOTE" "$BASE" "$MERGED"

The following tools are valid, but not currently available:
    araxis
    tortoisemerge
    winmerge

Some of the tools listed above only work in a windowed
environment. If run in a terminal-only session, they will fail.
```

Графический инструмент разрешения конфликта (продолжение)



Графический инструмент разрешения конфликта (продолжение)

Пример настройки `p4merge` в качестве графического инструмента разрешения конфликтов:

```
$ git config merge.tool p4merge
$ git config mergetool.p4merge.cmd \
  "'C:/Program Files/Perforce/p4merge.exe' \$BASE \$LOCAL \$REMOTE \$MERGED"
$ git config mergetool.prompt false
$ git config mergetool.trustExitCode false
$ git config mergetool.keepBackup false
```

☝️ В последних версиях Git этого не требуется для большинства программ.

Графический инструмент разрешения конфликта (продолжение)

```
$ git mergetool [-t <tool>] [file]
```

```
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge p4merge araxis bc3 codecompare vimdiff emerge
Merging:
index.html

Normal merge conflict for 'index.html':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (opendiff):
```

Завершение конфликта слияния

```
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:   index.html
```

Пример сообщения по умолчанию

```
Merge branch 'testing'

Conflicts:
    index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#   .git/MERGE_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#   modified:   index.html
#
```

🤔 При слиянии ничего не происходит?

- Вливаемые изменения уже присутствуют в целевой ветке.
- Поэтому некоторые коммиты при перебазировании могут пропасть.

🤔 А если конфликтов нет, то всё хорошо?

🤔 Я очень боюсь конфликтов слияние, поэтому буду заливать свои изменения первым!

Практика 💪

- Изменить один файл в ветках hotfix и master. Получить conflict слияния и разрешить его.

Заключение

Ваши вопросы ?