**32. Разрешение конфликтов слияния в СКВ Git.**

**Отмена слияния**

Если вы не смогли слить ветви и решили просто отказаться и отказаться, вы всегда можете вернуться к состоянию до слияния :

$ git reset --hard HEAD

Или, если вы уже зафиксировали слияние коммитом, что вы хотите его отменить,

$ git reset --hard ORIG\_HEAD

Однако эта последняя команда может быть опасной в некоторых случаях когда ваш коммит имеет уже коммиты потомки, или если этот коммит сам является слиянием с другой ветвью, тогда удаление коммита запутает будущие слияния.

**Быстрое слияние вперёд**

Существует один особый случай, не упомянутый выше, которая третируется по-разному. Нормально, результатом слияния является коммит слияния с двумя родителями, в точке объединения на двух линий разработки.

Однако, если текущая ветка является потомком другой и каждый коммит всегда один и всегда содержат коммит предыдущего, то можно просто выполнив команду git`ом “fast forward” (Быстрое движение вперёд) вершина текущей ветки продвинется вперед до точки вершины объединения в новую ветвь, без каких-либо новых созданий коммитов.

**Исправление ошибок**

Если вы сделали ошибку в рабочем каталоге, но пока еще не зафиксировали вашу ошибку, вы можете вернуть рабочий каталог в прошлое состояние командой

$ git reset --hard HEAD

Если вы сделаете коммит и позже вы решили отказаться от него, то у есть два принципиально различных способа решения проблемы:

1. Вы можете создавать новые коммит, в котором отменить всё, что вы сделали в старом коммите. Это правильный вариант, если ваши ошибки уже были преданы гласности.
2. Ни в коем случае не делайте этого, если вы уже сделали историю гласной; git обычно не ожидает изменения истоии проекта, и не может корректно выполнять частые слияния с веткой, у которой имело изменение истории. Напрример, от удаляемого вами коммита кто-то уже мог создать отдельную ветку разработки и такая ветка не сможет быть в дальнейшем корректно слита с вашим проектом.

**Исправление ошибки новым коммитом**

Создание нового коммита, который отменяет ранее сделанные исправления, очень легко, просто дайте команду git-revert(1) с ссылкой на плохой коммит; например, вернуться к последниме коммиту:

$ git revert HEAD

Это позволит создать новыЙ коммит, который “откатывает” изменения HEAD. Вам будет предоставлена возможность отредактировать сообщение для совершения нового коммита.

Вы также можете вернуть последние изменения, например, предпоследние:

$ git revert HEAD^

В этом случае git попытается отменить старые изменения, оставляя нетронутыми любые изменения, сделанные после коммита.

Если последние изменения пересекаются с изменениями будет восстановления, Вам будет предложено устранить конфликты вручную, как и в случае удалени конфликтов a merge .

**Исправление ошибки путём переписывания истории**

Если взять проблематичный коммит является последним коммитом, и вы еще не сделали его публичным, вы можете просто удалить его, используя git-reset .

Кроме того, вы можете отредактироватьсодержимое рабочего каталога и обновлять индекс для устранения ошибки, так-же, как если бы вы хотели создать новый коммит, а запустив

$ git commit --amend

который заменит старый коммит новым путём включения ваших изменений, и давая вам возможность изменить вначале старое сообщение коммита.  
Опять же, вы никогда не должны делать это с коммитом который уже был объединён в другую ветвь, в этом случае лучше использовать git-revert(1).  
Кроме того, можно заменить более старые коммиты в истории, но это более сложная тема и мы оставим её его для другой главы.

**33. Работа с удаленными репозиториями в СКВ.**

Работа с удалёнными репозиториями

Для того, чтобы внести вклад в какой-либо Git-проект, вам необходимо уметь работать с удалёнными репозиториями. Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети. У вас может быть несколько удалённых репозиториев, каждый из которых может быть доступен для чтения или для чтения-записи. Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и получение данных из них. Управление репозиториями включает в себя как умение добавлять новые, так и умение удалять устаревшие репозитории, а также умение управлять различными удалёнными ветками, объявлять их отслеживаемыми или нет и так далее.