При работе с системой контроля версий, такой как Git, конфликты возникают, когда два человека одновременно изменяют одну и ту же часть одного и того же файла. Для автоматического разрешения таких конфликтов можно настроить стратегию слияния.

1. **Создание файла .gitattributes:** Файл .gitattributes позволяет указать Git, какие стратегии использовать для определенных типов файлов. Создаем этот файл в корне нашего проекта:

touch .gitattributes

1. **Добавляем правила слияния:** Открываем файл .gitattributes и добавляем следующие строки:
2. \*.txt merge=ours
3. \*.md merge=theirs
4. README.md merge=union
5. \* merge=recursive
6. ```.
7. Эти настройки означают следующее:
8. - `\*.txt merge=ours`: Для всех файлов с расширением `.txt` будет использоваться версия из текущей ветки (`ours`), если возникнет конфликт.
9. - `\*.md merge=theirs`: Для всех файлов с расширением `.md` будет использоваться версия из другой ветки (`theirs`) при конфликтах.
10. - `README.md merge=union`: Для конкретного файла `README.md` будет применяться стратегия объединения (`union`), то есть обе версии будут объединены в одном файле.

- `\* merge=recursive`: Для остальных файлов будет использоваться стандартная рекурсивная стратегия слияния (`recursive`).

1. **Проверяем работу системы:**Теперь создадим сценарий, где мы попытаемся изменить один и тот же файл с двух разных веток и посмотрим, как сработают наши правила.

**Шаг 1: Создаем две ветки и вносим изменения**

Предположим, у нас уже есть основной файл example.txt.

* 1. Переходим на новую ветку:

git checkout -b feature-a

* 1. Изменяем содержимое файла example.txt: Добавляем строку "This is a change from feature-a" и сохраняем файл.
  2. Фиксируем изменения:
  3. git add example.txt

git commit -m "Change in feature-a"

* 1. Возвращаемся на главную ветку (master):

git checkout master

* 1. Создаем еще одну ветку для второй группы изменений:

git checkout -b feature-b

* 1. Опять меняем содержимое файла example.txt, теперь добавляем строку "This is another change from feature-b" и фиксируем изменения аналогично предыдущему шагу:
  2. git add example.txt

git commit -m "Another change in feature-b"

**Шаг 2: Пробуем слить ветки**

Попробуем объединить изменения из ветки feature-b обратно в master:

git checkout master

git merge feature-b

Поскольку у нас настроена стратегия слияний, Git автоматически применяет соответствующие правила. Например, для файла example.txt, который имеет расширение .txt, будет использована стратегия merge=ours, поэтому изменения из feature-a останутся нетронутыми.

Если бы мы попытались сделать то же самое с файлом README.md, который использует стратегию merge=union, то Git объединил бы оба изменения в этом файле, сохранив обе строки.

**Проверка результатов**

После выполнения слияния можно проверить состояние файлов:

1. **Содержимое example.txt:** Оно должно содержать только изменения из первой ветки (feature-a), поскольку мы использовали стратегию ours.
2. **Если был изменен README.md:** Он должен содержать объединенные изменения из обеих веток, так как для него была указана стратегия union.

Таким образом, наша настройка стратегий слияния позволила автоматизировать разрешение некоторых конфликтов, минимизируя необходимость ручного вмешательства.