

Git merge (브랜치 병합) 하는 방법

Git을 버전관리시스템으로 두고 같이 협업하는 과정에서 브랜치를 만들어서 각자 개발을 했다면,

언젠가는 소스코드를 합쳐서 테스트도 해보고 시스템에 적용도 해보고 해야한다.

그런 상황에서 브랜치를 합치는 것(병합)을 merge라 한다.

* 사용법

A브랜치에서 B브랜치를 병합한다 (= A로 B를 가져온다. A<-B)

: **git merge B브랜치명** (ex. git merge exp)

merge에는 크게 두 가지 상황이 존재한다.

1. 서로 다른 파일을 수정했을 때
2. 서로 같은 파일을 수정했을 때

먼저 1번의 상황은 불편할 것이 하나도 없다.

만약 A라는 사람이 A.java 파일을 수정했고 B라는 사람이 B.java 파일을 수정했을 때 merge 한다면 git이 자동으로 소스코드를 합쳐줄 것이다. (충돌 없음)

2번의 상황은 또 다시 세부적으로 두가지 경우로 나눌 수 있다.

- 2-1) 서로 같은 이름의 파일을 수정했지만 수정한 부분이 다를 때
- 2-2) 서로 같은 이름의 파일을 수정하고 수정한 부분이 겹칠 때

예를 들어 2-1)의 상황은 A라는 사람이 x라는 파일의 1~200번째 줄까지 수정했고 B라는 사람이 x라는 파일의 400~600번째 줄까지 수정했다면 이 역시도 1번 상황과 마찬가지로 git이 자동으로 소스코드를 합쳐줄 것이다.

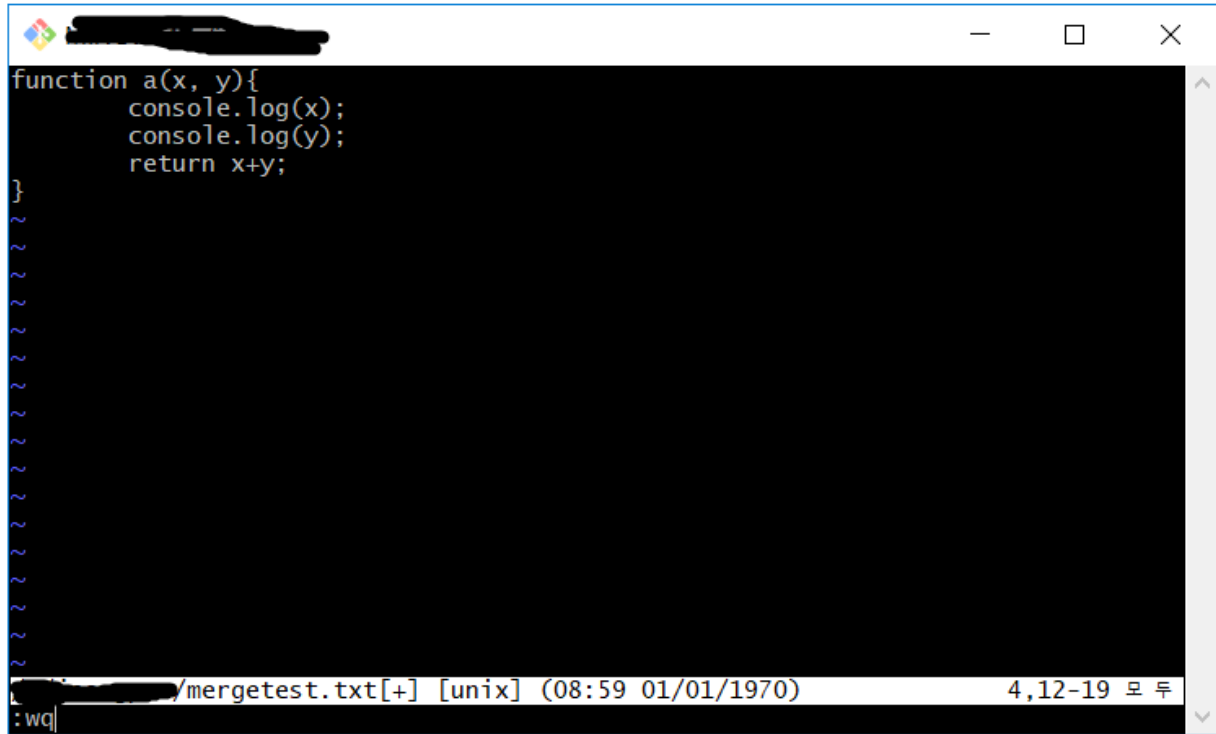
2-2)의 상황은 A라는 사람이 x라는 파일의 특정 메서드 f()의 특정 부분을 수정했다고 하고, B라는 사람이 똑같이 x라는 파일의 특정 메서드f()의 특정 부분을 수정했다면 (최소 한줄이라도 같은 부분을 수정했다고 가정) git은 자동으로 merge를 해줄 수 없는 상황이라고 사용자에게 알리고 사용자가 그 부분을 수정해주기를 위임한다.

이런 상황을 **conflict** 충돌이 일어났다고 한다.

git이 auto merge를 해주는 상황에서는 할말이 없다.

충돌나는 부분이 없으니 훌륭하게 git이 처리해주었기 때문이다.

그러나 2-2) 와 같은 상황처럼 충돌이 일어나면 개발자가 직접 해당 부분의 소스코드를 고쳐주어야 한다.



The image shows a screenshot of a code editor window. The editor has a dark background with light-colored text. The code visible is a JavaScript function definition:

```
function a(x, y){  
  console.log(x);  
  console.log(y);  
  return x+y;  
}
```

Below the code, there is a status bar with the following text: `/mergetest.txt[+] [unix] (08:59 01/01/1970) 4,12-19 모두`. The status bar also shows a cursor position `:wq|` on the left.

exp브랜치에서는 아래와 같이 소스를 수정하고 commit을 했고

master브랜치에서는 아래와 같이 소스를 수정하고 commit을 했다.

이 상황에서 master브랜치에서 exp브랜치를 merge하기 위해 "git merge exp" 라는 명령어를 치면 어떨까?

특히 conflict가 발생한 파일을 열어보면 쉽게 알 수 있다.

[illegible]

보면 <<<<< HEAD 부터 >>>>>까지가 충돌이 일어난 부분이고

<<<<< HEAD 부터 =====까지는 master브랜치의 소스코드고

=====부터 >>>>> exp까지는 exp브랜치의 소스코드다.

git이 이 부분을 알려주니 그 부분의 코드를 잘 수정해서 파일을 저장한 후 다시 commit을 해주면 올바르게 merge가 된다.

* merge에서 메시지 파악하기

1. **fast-foward** : 빨리감기라는 뜻으로 예를 들어 A브랜치에서 급하게 다른 것을 만들어보려고 B브랜치를 생성하고 B브랜치는 여러 번의 commit이 있었다.

이런 상황에서 A브랜치에서 B브랜치를 merge한다고 가정할 때, A브랜치에서 B브랜치를 분기해준 것 외에 별 다른 commit이 없었으므로 A브랜치는 B브랜치만큼 그냥 "빨리감기"하면 되는 상황이다.

그래서 이런 상황일 때 별도의 commit log없이 B브랜치가 만든 최신의 commit log를 가리키게 만드는 것을 fast-foward라고 한다.

2. **recursive strategy** : 재귀적인 전략으로 예를 들어 A브랜치에서 B브랜치를 생성했다. 그 후 A브랜치도 commit이 여러번 있었고 B브랜치도 commit이 여러번 있었을 때 A브랜치에서 B브랜치를 merge한다.

이러면 fast-foward할 수 없고 git은 A브랜치와 B브랜치의 조상 commit을 찾아 내부적으로 3way merge라는 방법을 이용하여 merge를 해준다.