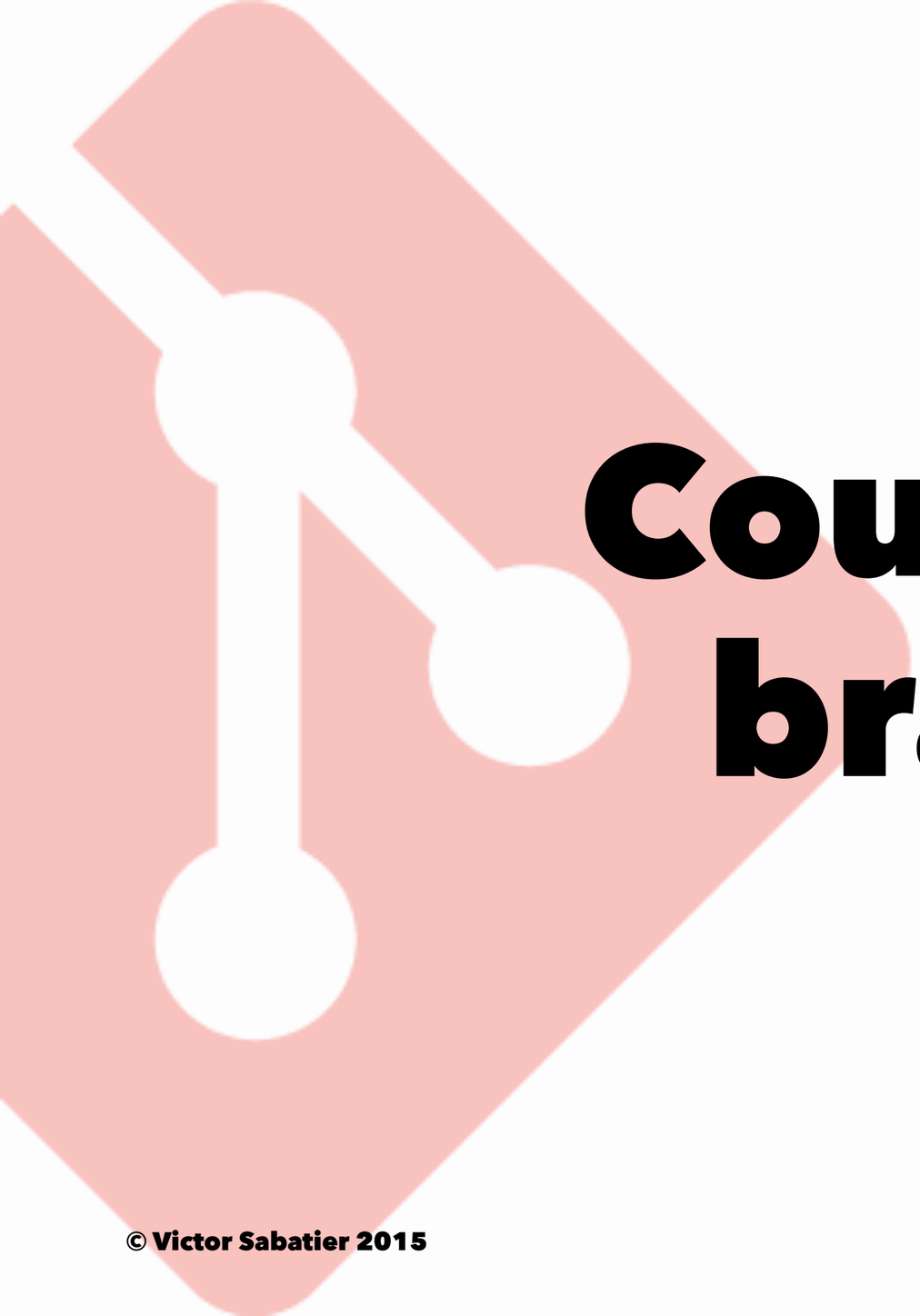
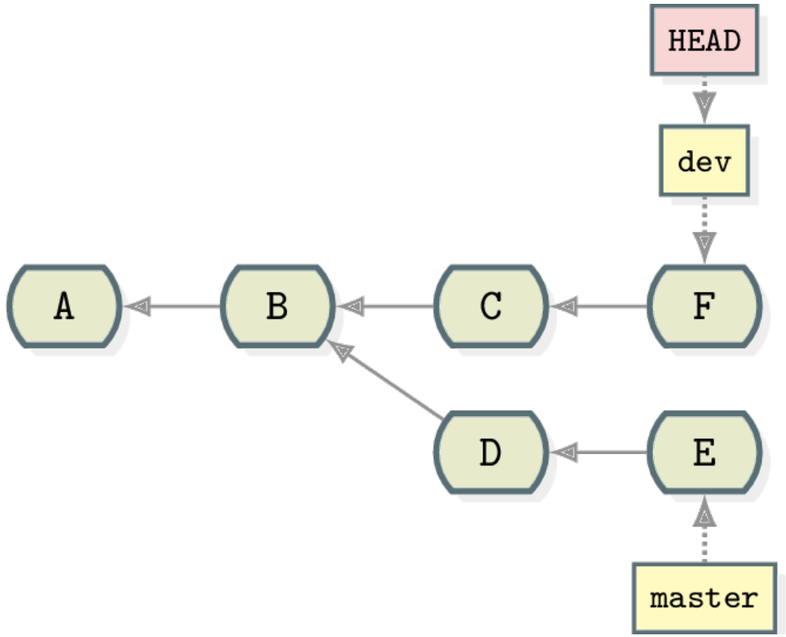


Cours 3 : Les branches.

Une branche est un pointeur sur un commit[1]



[1] Un commit appartient à une branche si il est accessible depuis cette branche
Lors d'un **git init**, on a une branche par défault qui s'appelle master.

Faire une branche permet de diverger de la ligne principale de développement sans l'impacter.

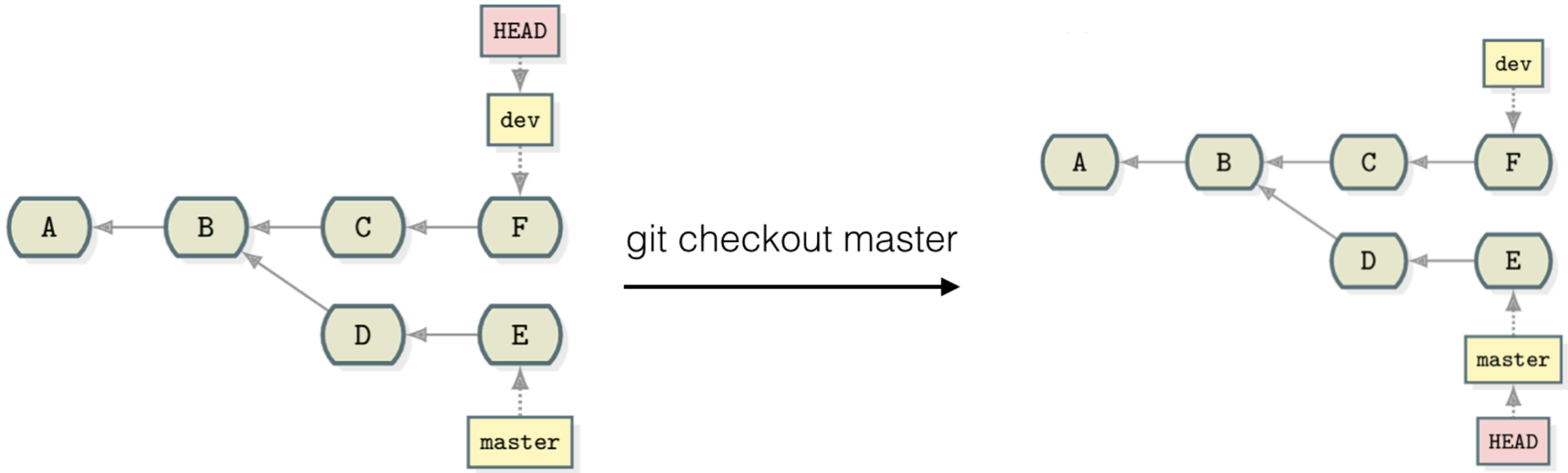
Lister les branches :

```
1 | git branch [-r | -a]
```

```
➜  Realtime-Markdown-Viewer git:(master) ✗ git branch -a
  code
* master
  new_line
  parser_v1
  tp4
  tp5
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/new_line
  remotes/origin/parser_v1
```

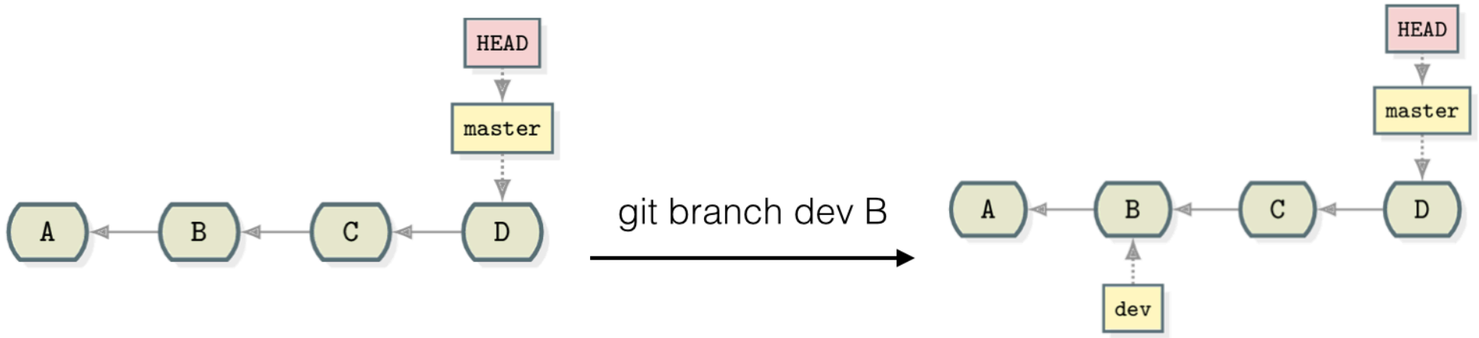
Changer de branches :

```
1 | git checkout BRANCH
```



Créer une branche[2] :

```
1 | git branch BRANCH [REF]
```



[2] Une référence peut être, soit un SHA1, soit le nom d'une branche, soit un tag, soit un raccourci du type HEAD^ (parent du commit pointé par HEAD).

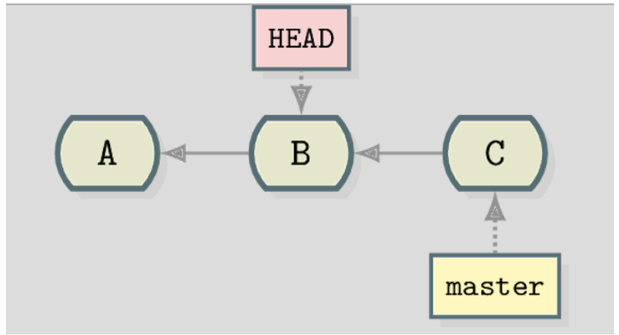
Créer une branche pour en changer :

```
1 | git checkout -b BRANCH [REF]
```

Supprimer une branche :

```
1 | git branch -d BRANCH
```

Detached HEAD.



Checkout sur un commit et pas sur une étiquette :

```
➜  Realtime-Markdown-Viewer git:(master) ✗ git checkout c3af8ef
Note: checking out 'c3af8ef'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
```

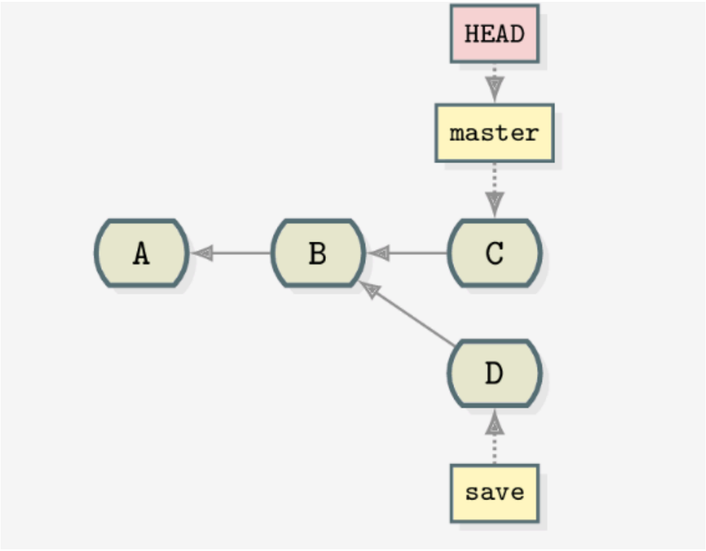
Commit orphelin

D n'appartient à aucune branche : il est orphelin.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

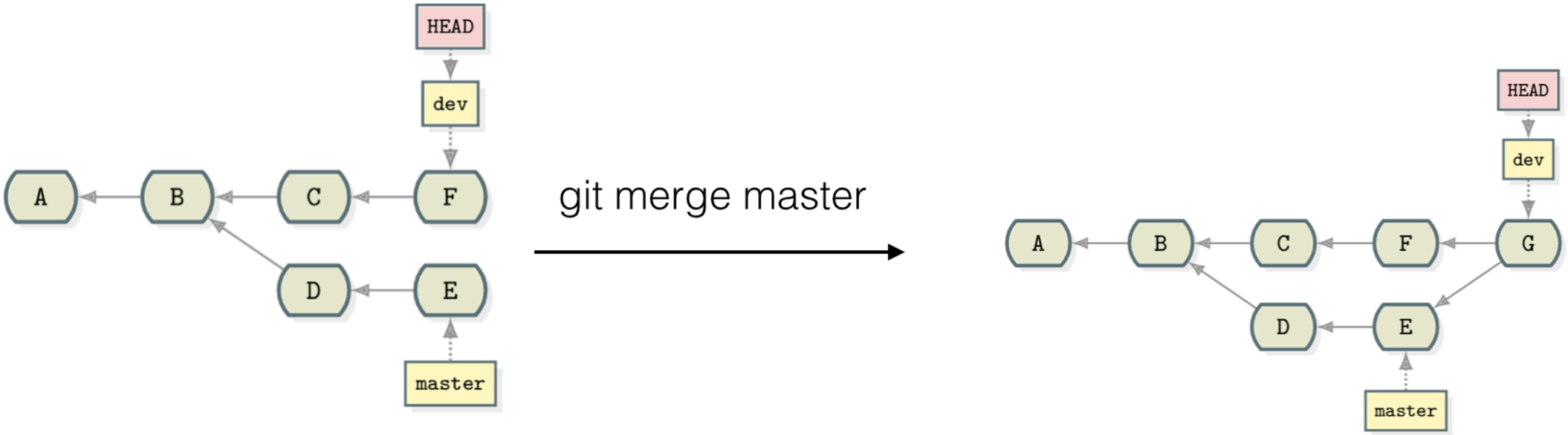
```
git checkout -b new_branch_name
```

```
1 | git branch save D
```

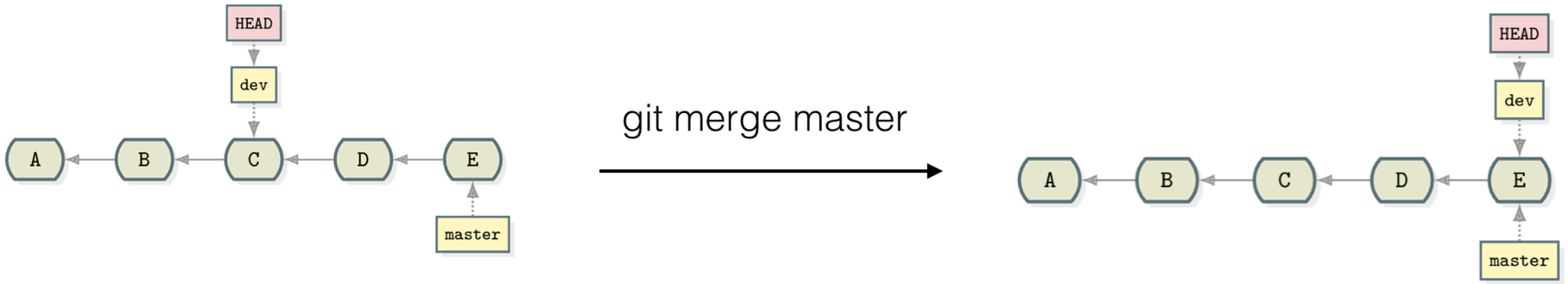


Merge

```
1 | git merge BRANCH
```



Le merge fast forward




```
➜  Realtime-Markdown-Viewer git:(tp4) ✗ git merge tp5
Updating c853da9..b496d8c
Fast-forward
```

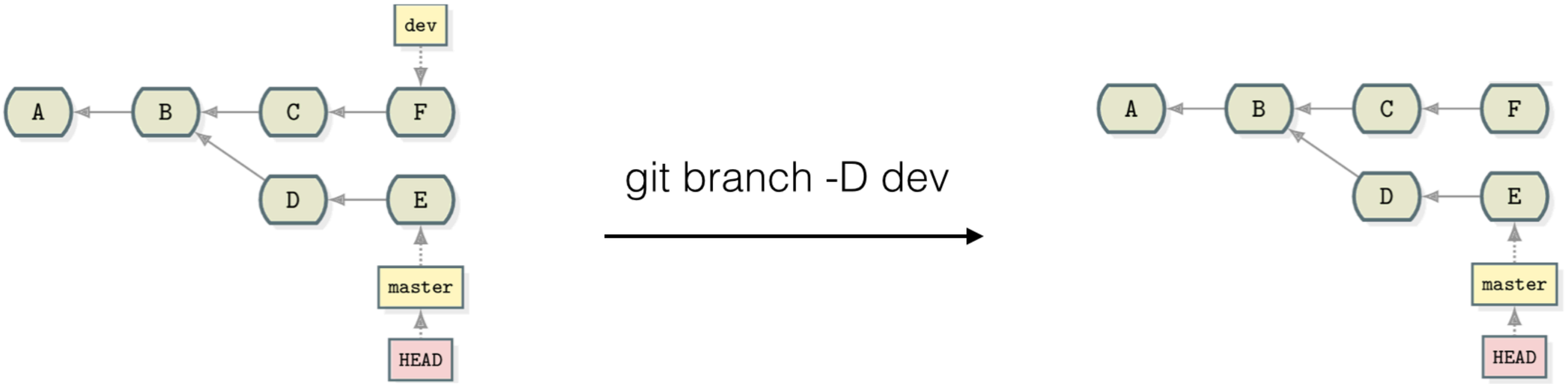
Supprimer une branche non mergée

Git refuse par défaut de supprimer une branche non mergée :

```
➜  Realtime-Markdown-Viewer git:(master) ✗ git branch -d code
error: The branch 'code' is not fully merged.
If you are sure you want to delete it, run 'git branch -D code'.
```

Pour le forcer :

```
1 | git branch -D BRANCH
```



Les commits C et F deviennent orphelins.

Les conflits

L'opération de merge entraine parfois des conflits.

```
$ git merge tp3
Auto-merging converter/markdown.js
CONFLICT (content): Merge conflict in converter/markdown.js
Automatic merge failed; fix conflicts and then commit the result.
```

On repère la zone en conflit dans le fichier avec les marqueurs de merge :

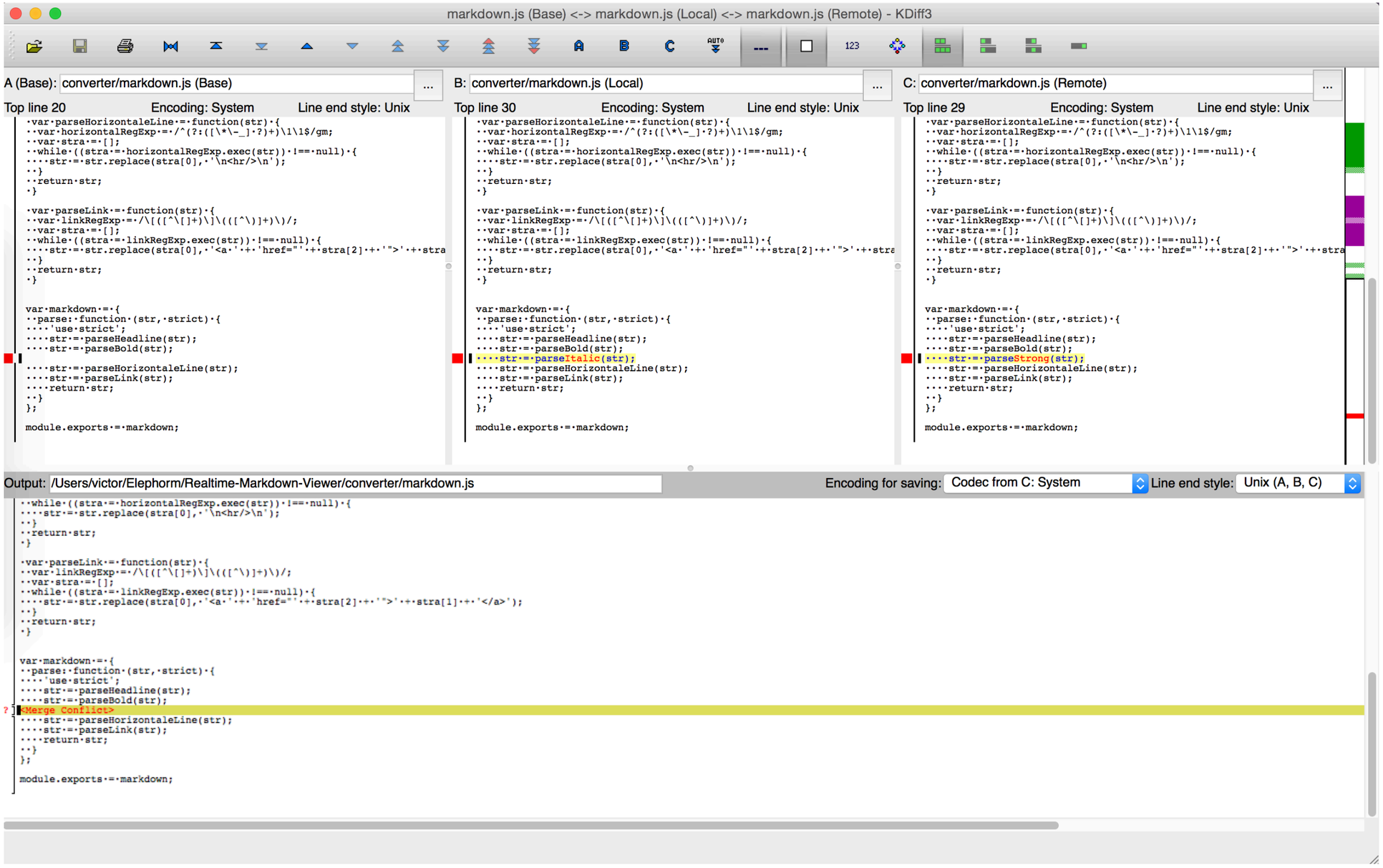
```
      str = parseHeadline(str);
      str = parseBold(str);
++<<<<<<< HEAD
 +    str = parseItalic(str);
++=======
+     str = parseStrong(str);
++>>>>>>> tp3
      str = parseHorizontaleLine(str);
      str = parseLink(str);
      return str;
```

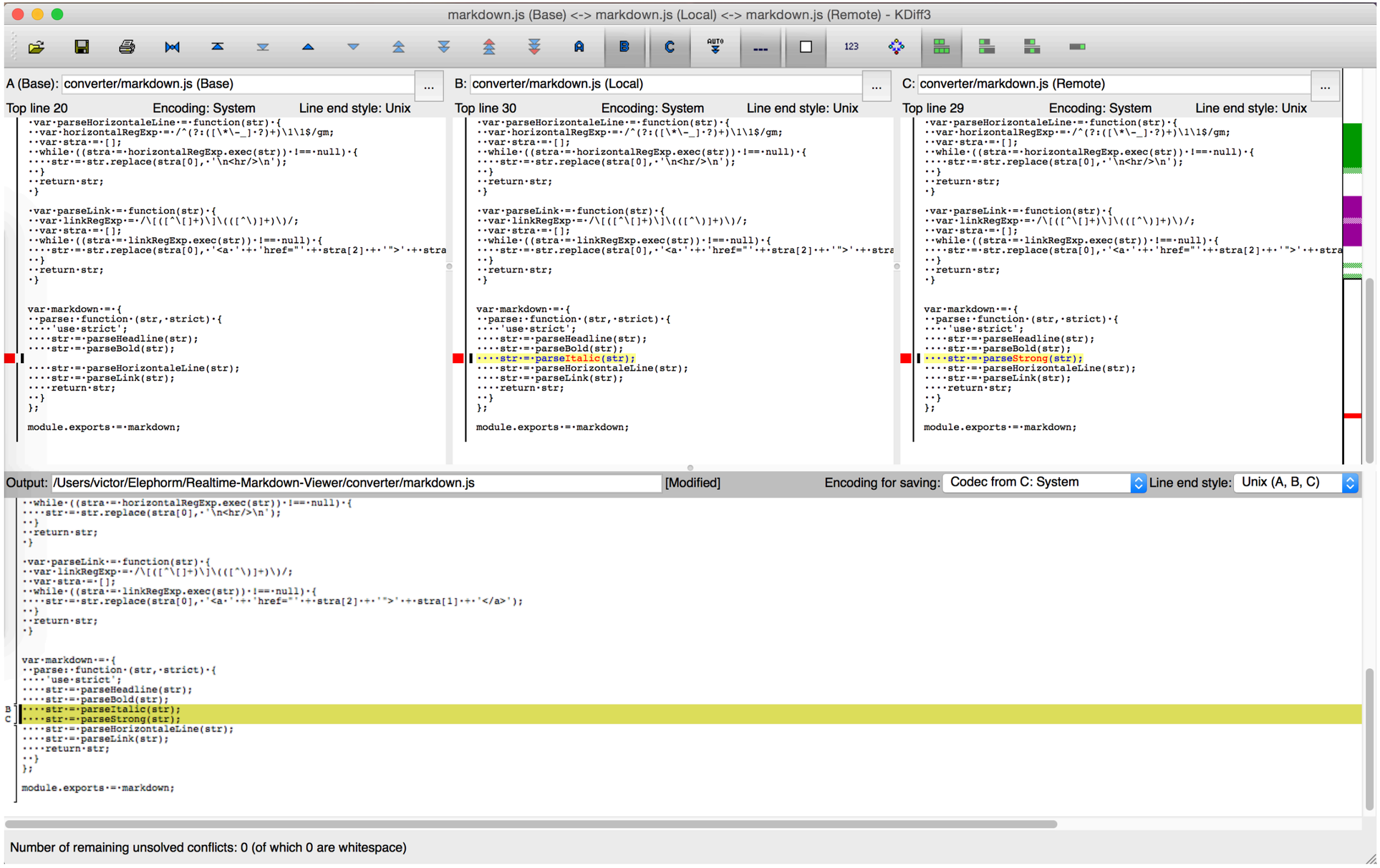
La gestion de conflits : méthode manuelle.

1. Ouvrir les fichiers en conflit.
2. Résoudre le conflit en sélectionnant les parties désirées.
3. **Ajouter les fichiers à la staging area.**
4. Faire le commit de merge.

La gestion de conflits : utiliser un mergetool.

1. Ouvrir votre mergetool.
2. Résoudre les conflits.
3. Sauvergarder la résolution.
4. Faire un commit de merge.




markdown.js (Base) <-> markdown.js (Local) <-> markdown.js (Remote) - KDiff3
A (Base): converter/markdown.js (Base)
B: converter/markdown.js (Local)
C: converter/markdown.js (Remote)
Top line 20 Encoding: System Line end style: Unix
Top line 30 Encoding: System Line end style: Unix
Top line 29 Encoding: System Line end style: Unix
Output: /Users/victor/Elephorm/Realtime-Markdown-Viewer/converter/markdown.js
[Modified]
Encoding for saving: Codec from C: System
Line end style: Unix (A, B, C)
Number of remaining unsolved conflicts: 0 (of which 0 are whitespace)