人生不如意之事十之八九，合并分支往往也不是一帆风顺的。

准备新的 `feature1` 分支，继续我们的新分支开发：

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git switch -c feature1
Switched to a new branch 'feature1'

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (feature1)
$ git commit -m "AND simple"
[feature1 2fe9342] AND simple
 1 file changed, 1 insertion(+), 1 deletion(-)
```

切换到 `master` 分支：

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (feature1)
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

在 `master` 分支上把 `readme.txt` 文件的最后一行改为不一样的内容

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git add readme.txt

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git commit -m "& simple"
[master 434a088] & simple
 1 file changed, 1 insertion(+), 1 deletion(-)
```

这种情况下，Git 无法执行“快速合并”，只能试图把各自的修改合并起来，

但这种合并就可能会有冲突，我们试试看：

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git merge feature1
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

```
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   readme.txt
```

Git用`<<<<<<<`，`=======`，`>>>>>>>`标记出不同分支的内容，我们修改后保存：

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master|MERGING)
$ cat readme.txt
Git is a distributed version control system
Git is free software distributed under the GPL
Git has a mutable index called stage.
Git tracks changes of files.
<<<<<<< HEAD
Greating a new branch is quick & simple.
=======
Greating a new branch is quick AND simple.
>>>>>>> feature1

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master|MERGING)
$ git add readme.txt

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master|MERGING)
$ git commit -m "conflict fixed"
[master 1646b4c] conflict fixed


admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git log --graph --pretty=oneline --abbrev-commit
*   1646b4c (HEAD -> master) conflict fixed
|\
| * 2fe9342 (feature1) AND simple
* | 434a088 & simple
|/
* a03f09d (origin/master, origin/HEAD) restart all unfinished work
* 06b4fce transfer to pdf and html
* 9d384ce add document for add commit reset branch log status
* 86b33dd branch test
```

```
* 2ee2e28 (origin/dev) add test.txt
* 9a16e16 of files
* e9498aa git tracks changes
* 4aba66f understand how stage works
* dd1fcdd append GPL
```

最后，删除 `feature1` 分支

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git branch -d feature1
Deleted branch feature1 (was 2fe9342).
```

通常，合并分支时，如果可能，Git 会用 `Fast forward` 模式，但这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用 `Fast forward` 模式，Git 就会在 merge 时生成一个新的 commit，这样，从分支历史上就可以看出分支信息。

下面我们实战一下 `--no-ff` 方式的 `git merge`：

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git branch
* master

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git switch -c dev
Switched to a new branch 'dev'

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (dev)
$ git add .

admin@zebin MINGW64 /d/BlueTestTry/bluetooth (dev)
$ git commit -m "add merge"
[dev 5ff1a19] add merge
 4 files changed, 1 insertion(+), 1 deletion(-)
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git merge --no-ff -m "merge with no-ff" dev
Merge made by the 'recursive' strategy.
 readme.txt
```

```
admin@zebin MINGW64 /d/BlueTestTry/bluetooth (master)
$ git log --graph --pretty=oneline --abbrev-commit
*   f051498 (HEAD -> master) merge with no-ff
|\
| * 5ff1a19 (dev) add merge
|/
*   1646b4c (origin/master, origin/HEAD) conflict fixed
|\
| * 2fe9342 AND simple
* | 434a088 & simple
|/
* a03f09d restart all unfinished work
* 06b4fce transfer to pdf and html
* 9d384ce add document for add commit reset branch log status
* 86b33dd branch test
* 2ee2e28 (origin/dev) add test.txt
* 9a16e16 of files
* e9498aa git tracks changes
* 4aba66f understand how stage works
* dd1fcdd append GPL
* 20c0b88 add distributed
* 2557423 wrote an new readme file
```