现在，假定你已经完全掌握了暂存区的概念。下面，我们要讨论的就是，为什么 Git 比其他版本控制系统设计得优秀，因为 Git 跟踪并管理的是修改，而非文件。

你会问，什么是修改？比如你新增了一行，这就是一个修改，删除了一行，也是一个修改，更改了某些字符，也是一个修改，删了一些又加了一些，也是一个修改，甚至创建一个新文件，也算一个修改。

为什么说 Git 管理的是修改，而不是文件呢？我们还是做实验。第一步，对 readme.txt 做一个修改，比如加一行内容：

```
$ 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.
```

然后，添加：

```
$ git add readme.txt

$ git status

# On branch master

# Changes to be committed:

#   (use "git reset HEAD <file>..." to unstage)

#

#       modified:   readme.txt

#
```

然后，再修改 readme.txt：

```
$ 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.
```

提交：

```
$ git commit -m "git tracks changes"

[master 519219b] git tracks changes

 1 file changed, 1 insertion(+)
```

提交后，再看看状态：

```
$ git status

On branch master

Changes not staged for commit:

  (use "git add <file>..." to update what will be committed)

  (use "git checkout -- <file>..." to discard changes in working directory)


    modified:   readme.txt


no changes added to commit (use "git add" and/or "git commit -a")
```

咦，怎么第二次的修改没有被提交？

别激动，我们回顾一下操作过程：

第一次修改 -> `git add` -> 第二次修改 -> `git commit`

你看，我们前面讲了，Git 管理的是修改，当你用 `git add` 命令后，在工作区的第一次修改被放入暂存区，准备提交，但是，在工作区的第二次修改并没有放入暂存区，所以，`git commit` 只负责把暂存区的修改提交了，也就是第一次的修改被提交了，第二次的修改不会被提交。

提交后，用 `git diff HEAD -- readme.txt` 命令可以查看工作区和版本库里面最新版本的区别：

```
$ git diff HEAD -- readme.txt

diff --git a/readme.txt b/readme.txt
```

```
index 76d770f..a9c5755 100644

--- a/readme.txt

+++ b/readme.txt

@@ -1,4 +1,4 @@

 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.

+Git tracks changes of files.
```

可见，第二次修改确实没有被提交。

那怎么提交第二次修改呢？你可以继续 `git add` 再 `git commit`，也可以别着急提交第一次修改，先 `git add` 第二次修改，再 `git commit`，就相当于把两次修改合并后一块提交了：

第一次修改 -> `git add` -> 第二次修改 -> `git add` -> `git commit`

好，现在，把第二次修改提交了，然后开始小结。

小结

现在，你又理解了 Git 是如何跟踪修改的，每次修改，如果不用 `git add` 到暂存区，那就不会加入到 `commit` 中。