

自然，你是不会犯错的。不过现在是凌晨两点，你正在赶一份工作报告，你在 `readme.txt` 中添加了一行：

```
$ 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.

My stupid boss still prefers SVN.
```

在你准备提交前，一杯咖啡起了作用，你猛然发现了 `stupid boss` 可能会让你丢掉这个月的奖金！

既然错误发现得很及时，就可以很容易地纠正它。你可以删掉最后一行，手动把文件恢复到上一个版本的状态。如果用 `git status` 查看一下：

```
$ 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")
```

你可以发现，Git 会告诉你，`git checkout -- file` 可以丢弃工作区的修改：

```
$ git checkout -- readme.txt
```

命令 `git checkout -- readme.txt` 意思就是，把 `readme.txt` 文件在工作区的修改全部撤销，这里有两种情况：

一种是 `readme.txt` 自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是 `readme.txt` 已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次 `git commit` 或 `git add` 时的状态。

现在，看看 `readme.txt` 的文件内容：

```
$ 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 of files.
```

文件内容果然复原了。

`git checkout -- file` 命令中的 `--` 很重要，没有 `--`，就变成了“切换到另一个分支”的命令，我们在后面的分支管理中会再次遇到 `git checkout` 命令。

现在假定是凌晨 3 点，你不但写了一些胡话，还 `git add` 到暂存区了：

```
$ 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.

My stupid boss still prefers SVN.

$ git add readme.txt
```

庆幸的是，在 `commit` 之前，你发现了这个问题。用 `git status` 查看一下，修改只是添加到了暂存区，还没有提交：

```
$ git status

On branch master
```

```
Changes to be committed:
```

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

```
modified:   readme.txt
```

Git 同样告诉我们，用命令 `git reset HEAD <file>` 可以把暂存区的修改撤销掉（unstage），重新放回工作区：

```
$ git reset HEAD readme.txt
```

```
Unstaged changes after reset:
```

```
M    readme.txt
```

`git reset` 命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用 `HEAD` 时，表示最新的版本。

再用 `git status` 查看一下，现在暂存区是干净的，工作区有修改：

```
$ 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
```

还记得如何丢弃工作区的修改吗？

```
$ git checkout -- readme.txt
```

```
$ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

整个世界终于清静了！

现在，假设你不但改错了东西，还从暂存区提交到了版本库，怎么办呢？还记得[版本回退](#)一节吗？可以回退到上一个版本。不过，这是有条件的，就是你还没有把自己的本地版本库推送到远程。还记得Git是分布式版本控制系统吗？我们后面会讲到远程版本库，一旦你把 `stupid boss` 提交推送到远程版本库，你就真的惨了……

小结

又到了小结时间。

场景 1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令 `git checkout --file`。

场景 2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令 `git reset HEAD <file>`，就回到了场景 1，第二步按场景 1 操作。

场景 3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](#)一节，不过前提是没有推送到远程库。