前言

在上一笔记五中因为要处理紧急问题不得不将未完成的工作进行一次提交这样是有点不符合逻辑的，那难道就没有别的办法了吗？答案肯定是有的，我们在学习C语言中知道局部变量是存放再栈里面的，在Git中我们没有完成的工作也可以暂时存放在栈里面。

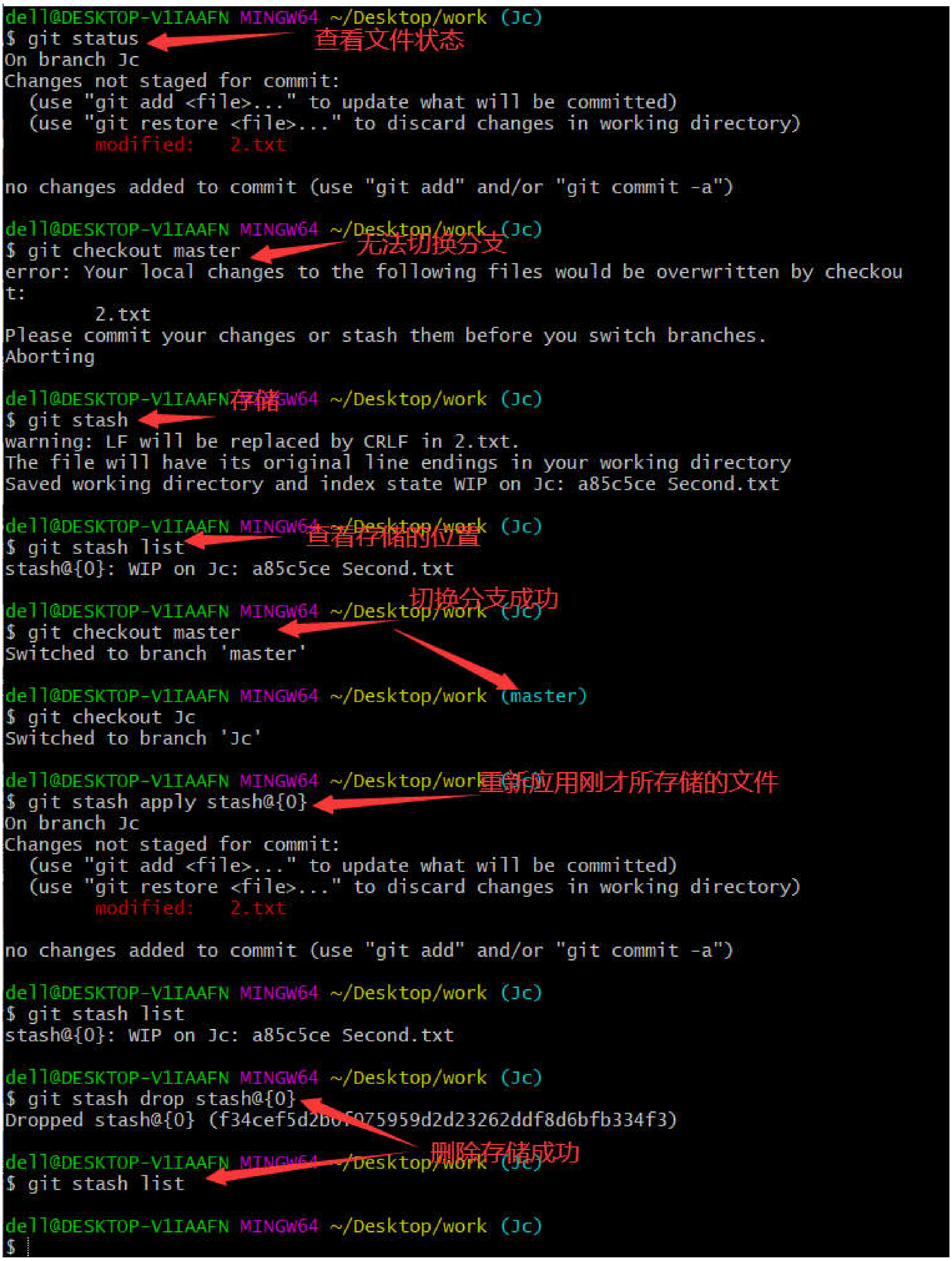
一、Git存储

不想因为短暂的离开此分支就给它进行一次提交，那么我们就应该好好的利用Git存储的命令。

命令

- git stash
  将未完成的修改放在一个栈上面。
- git stash list
  查看所存储的内容。
- git stash apply stash@{数字}
  重新应用在数字所指向的存储，但是这个命令并不会删除存储。
- git stash stash@{数字}
  删除在数字所指向的存储。
- git stash pop
  应用栈顶的存储并且删除它。

废话不多说，直接上图最明白了~



```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git checkout master
error: Your local changes to the following files would be overwritten by checkou
t:
        2.txt
Please commit your changes or stash them before you switch branches.
Aborting

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash
warning: LF will be replaced by CRLF in 2.txt.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on Jc: a85c5ce Second.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash list
stash@{0}: WIP on Jc: a85c5ce Second.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git checkout master
Switched to branch 'master'

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git checkout Jc
Switched to branch 'Jc'

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash apply stash@{0}
On branch Jc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash list
stash@{0}: WIP on Jc: a85c5ce Second.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash drop stash@{0}
Dropped stash@{0} (f34cef5d2b0f075959d2d23262ddf8d6bfb334f3)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git stash list

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$
```

总结：使用git statsh 真的没提交吗？答案肯定不提交了但是是Git给我自动提交的~

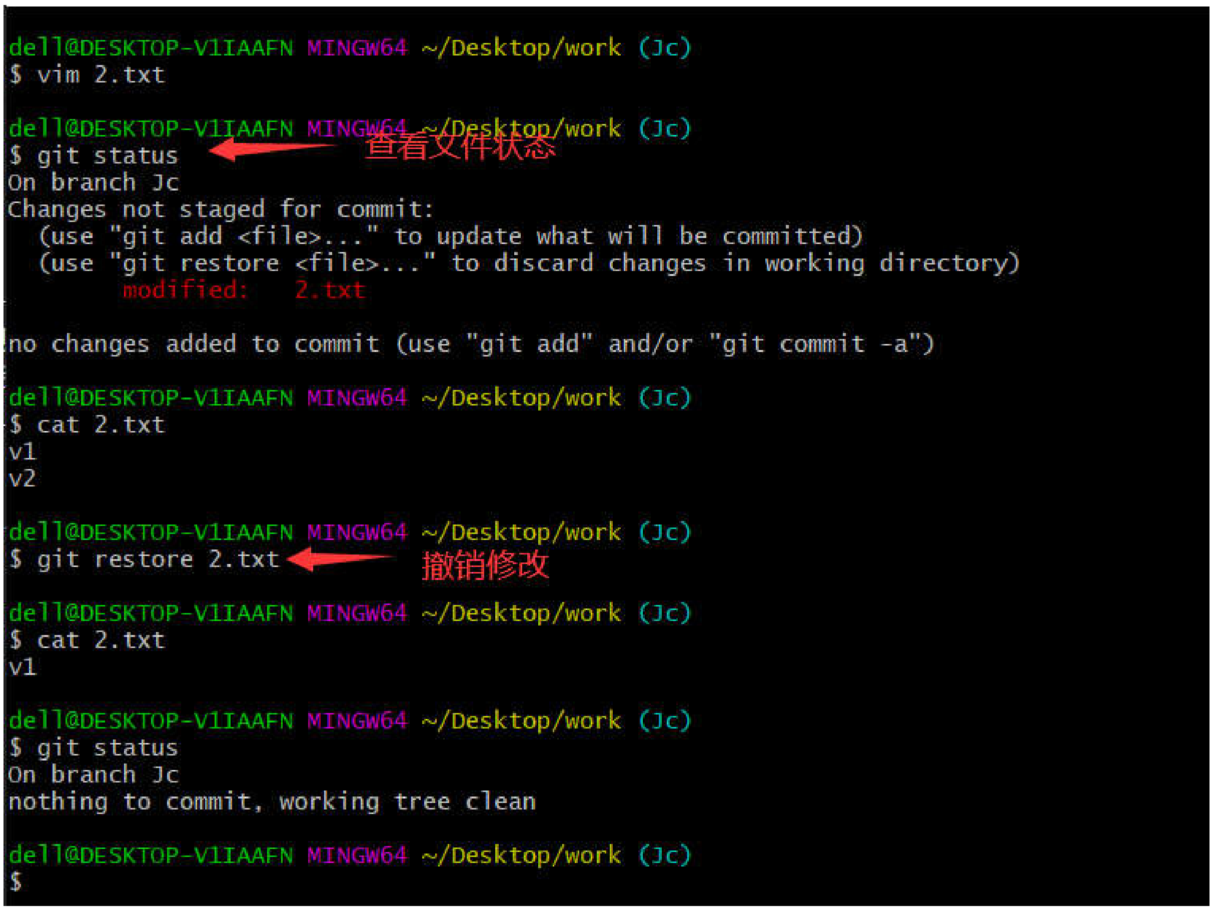
二、Git撤退

回顾一下我们所学的Git相关操作好像都是一股劲的往前冲冲冲！！没有撤销操作，所以嘞下面就看看撤销操作呗。首先我们所编写的代码无非就是在工作区、暂存区和版本库所以我们撤销操作也是围绕着这三个区域来进行的(前提是该文件已经被Git跟踪的哦)。

工作区的撤回

工作区撤回就是撤回在工作目录里面所作的修改(其实C-Z就可以实现哈哈哈)。

- 命令：git restore fileName
  上图演示

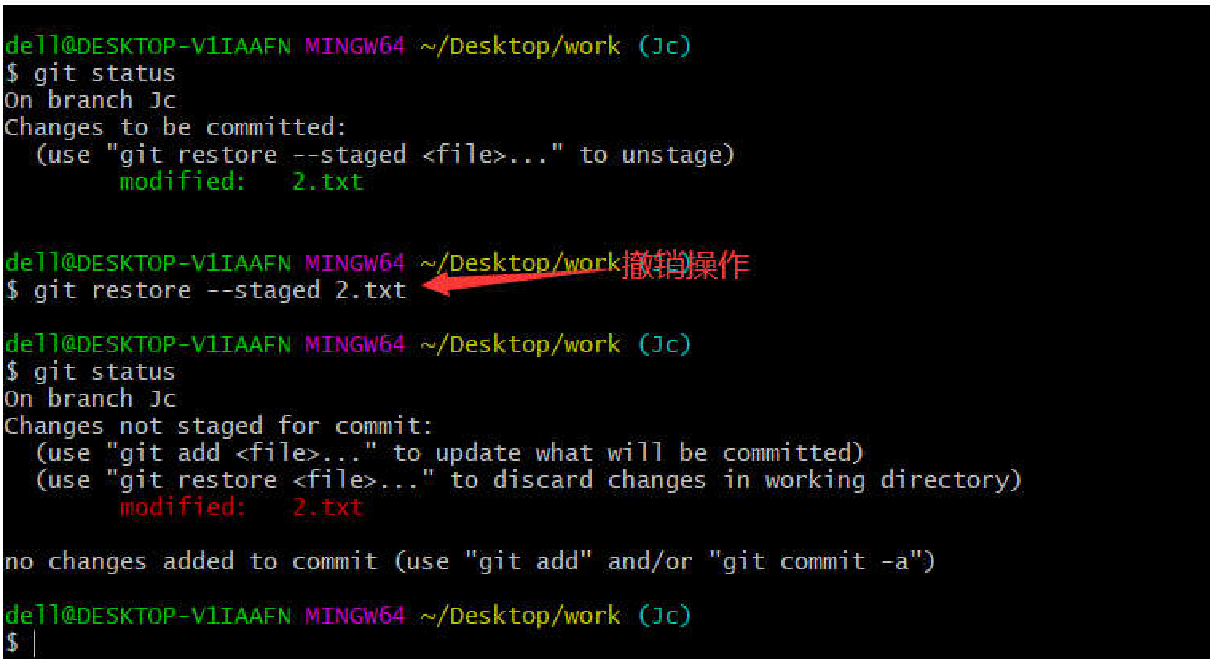


```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ vim 2.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ cat 2.txt
v1
v2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git restore 2.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ cat 2.txt
v1

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
nothing to commit, working tree clean

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$
```

暂存区的撤回

暂存区撤回就是撤回已经放入暂存区的内容。

- 命令：git restore --staged fileName
  上图演示

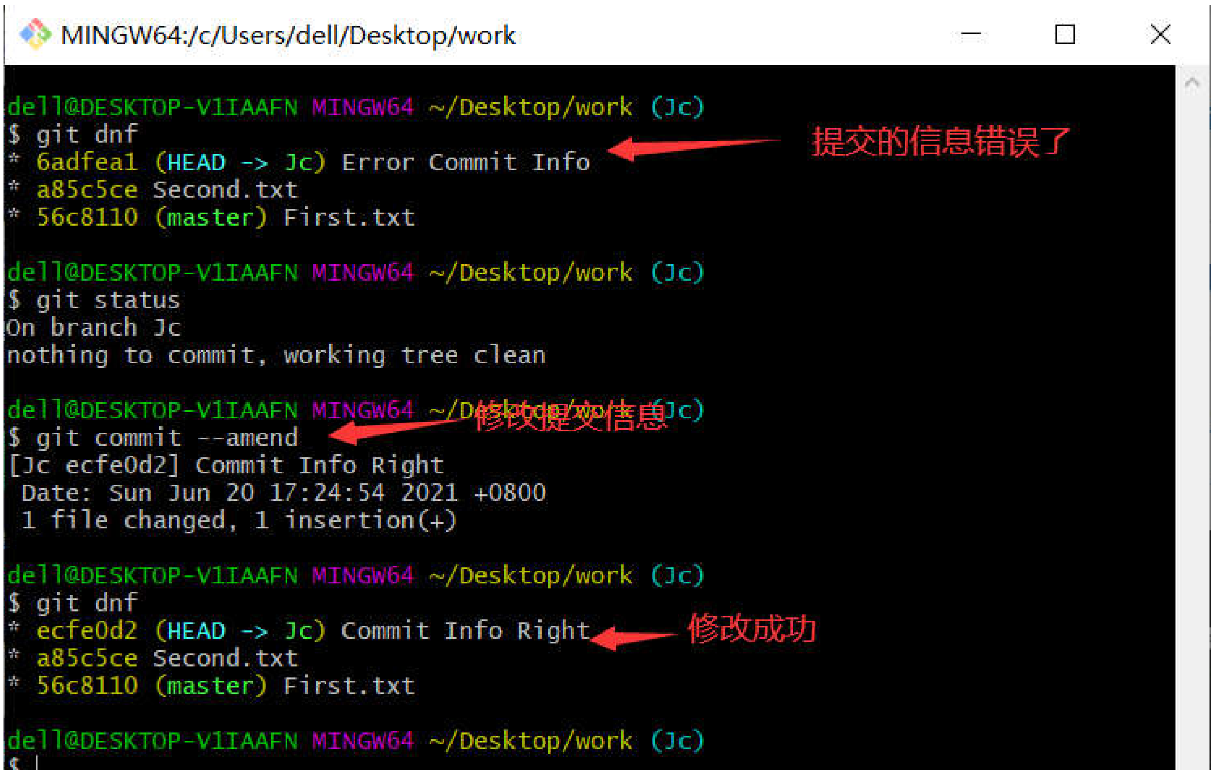


```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work
$ git restore --staged 2.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ |
```

版本库的撤回

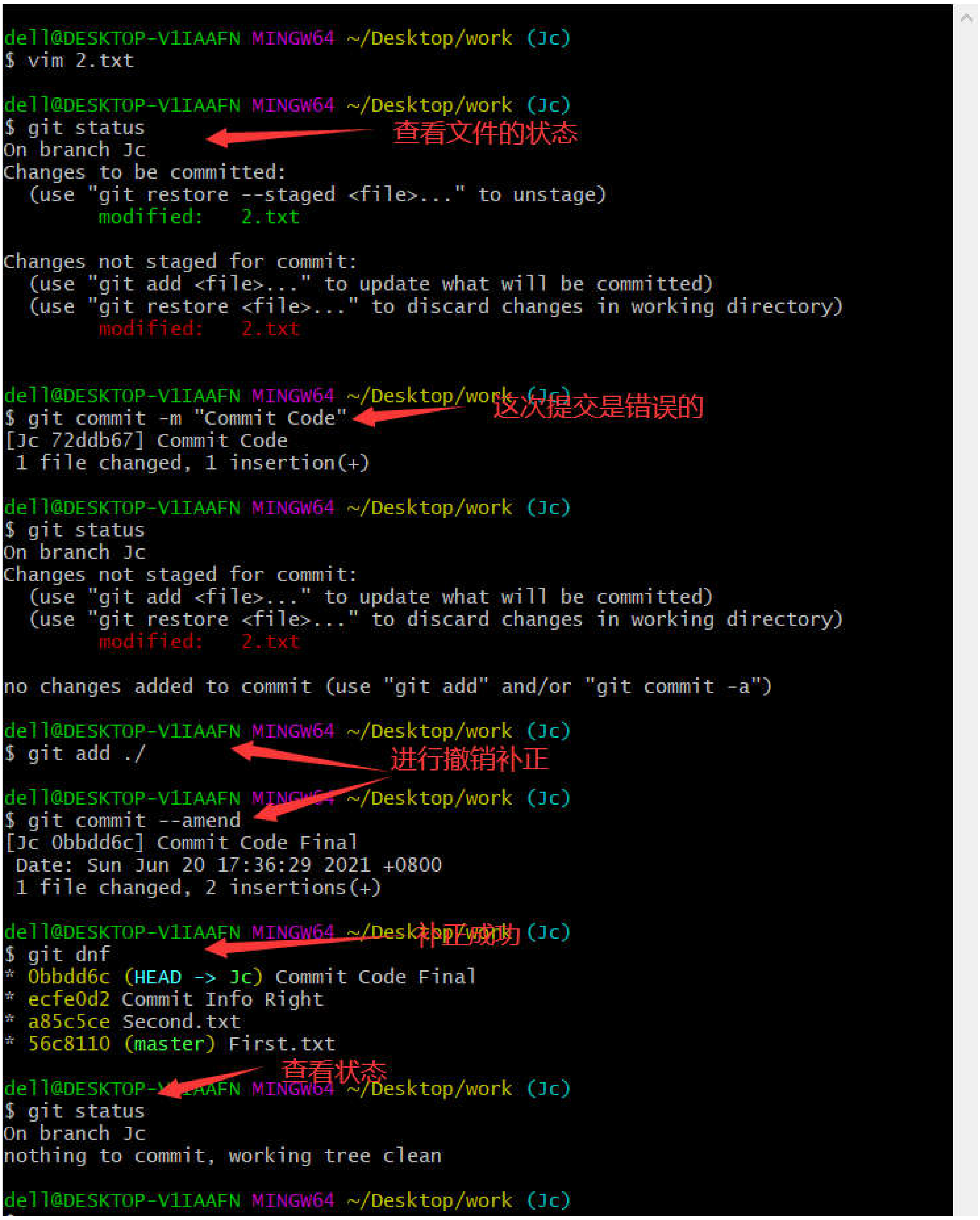
版本库撤回就是撤回已经提交的内容或者修改提交时候的所添加的信息。

- 命令：git commit --amend



```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git dnf
* 6adfea1 (HEAD -> Jc) Error Commit Info
* a85c5ce Second.txt
* 56c8110 (master) First.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
nothing to commit, working tree clean

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git commit --amend
[Jc ecfe0d2] Commit Info Right
 Date: Sun Jun 20 17:24:54 2021 +0800
 1 file changed, 1 insertion(+)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git dnf
* ecfe0d2 (HEAD -> Jc) Commit Info Right
* a85c5ce Second.txt
* 56c8110 (master) First.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$
```

除了提交信息有错误之外还是一种错误，例如：假设Jc.txt已经加入暂存区，此时我们修改了Jc.txt但是没有重新加入暂存区然后进行了提交操作，这样我们最后提交对象里面Jc.txt就会出现问题。第一个办法呢就是再次将Jc.txt 加入暂存区再进行提交写注释等，第二种办法就是利用git commit --amend命令，话不多说直接上图就完事了。



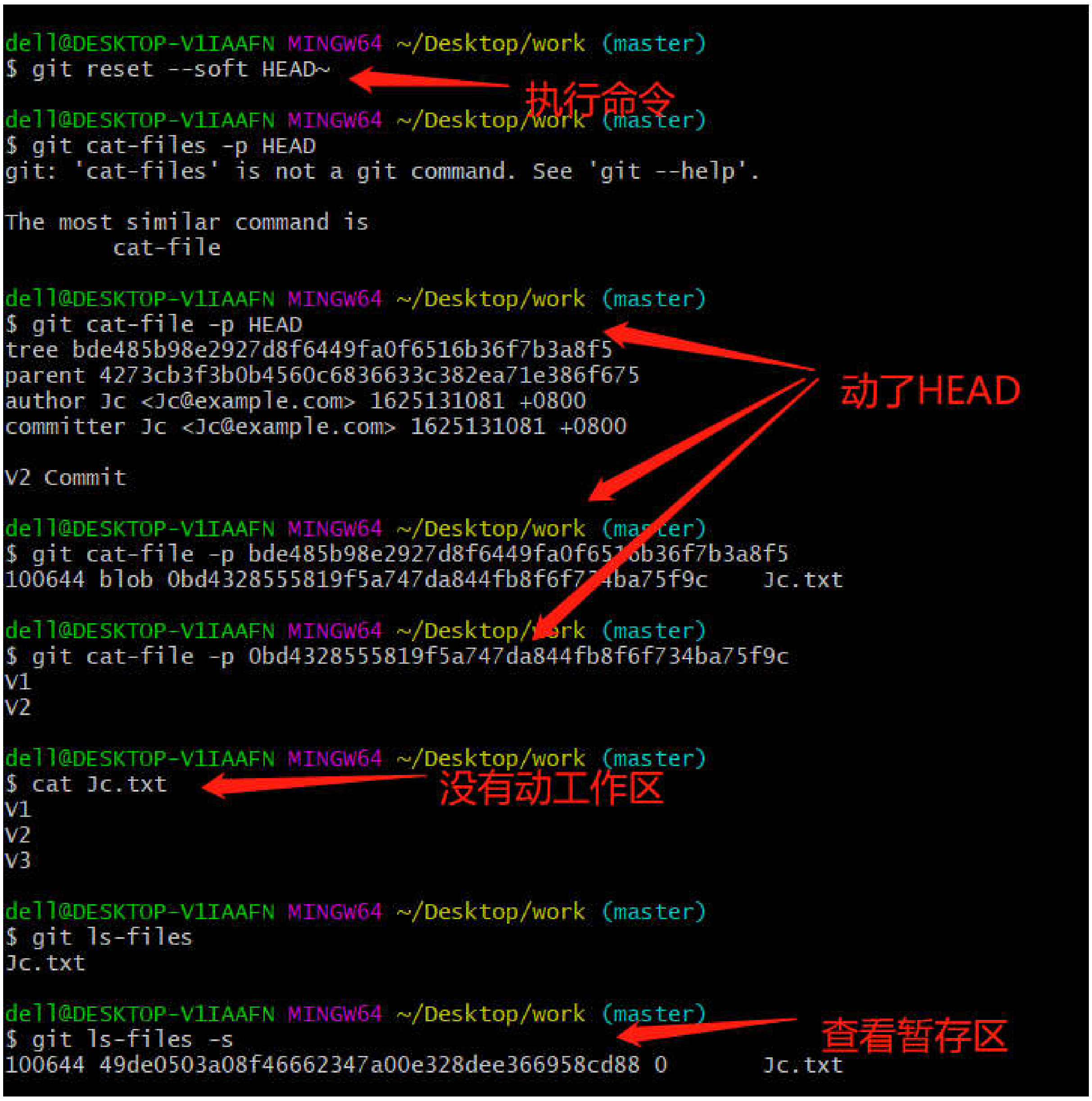
```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ vim 2.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt


dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git commit -m "Commit Code"
[Jc 72ddb67] Commit Code
 1 file changed, 1 insertion(+)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   2.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git add ./

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git commit --amend
[Jc 0bbdd6c] Commit Code Final
 Date: Sun Jun 20 17:36:29 2021 +0800
 1 file changed, 2 insertions(+)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git dnf
* 0bbdd6c (HEAD -> Jc) Commit Code Final
* ecfe0d2 Commit Info Right
* a85c5ce Second.txt
* 56c8110 (master) First.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
$ git status
On branch Jc
nothing to commit, working tree clean

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (Jc)
```

三、Git撤退的原理

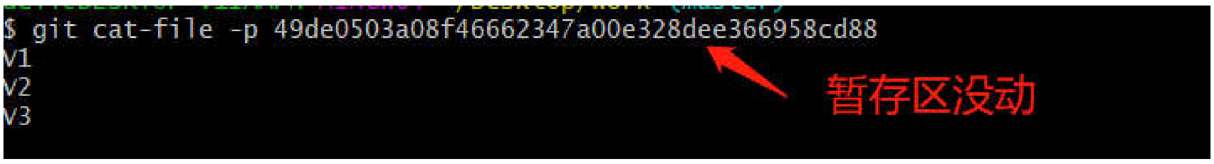
通过对上面的学习我们可以对Git撤退有一定的了解但是这是远远不够的，下面就是其原理进行解析。

reset相关命令

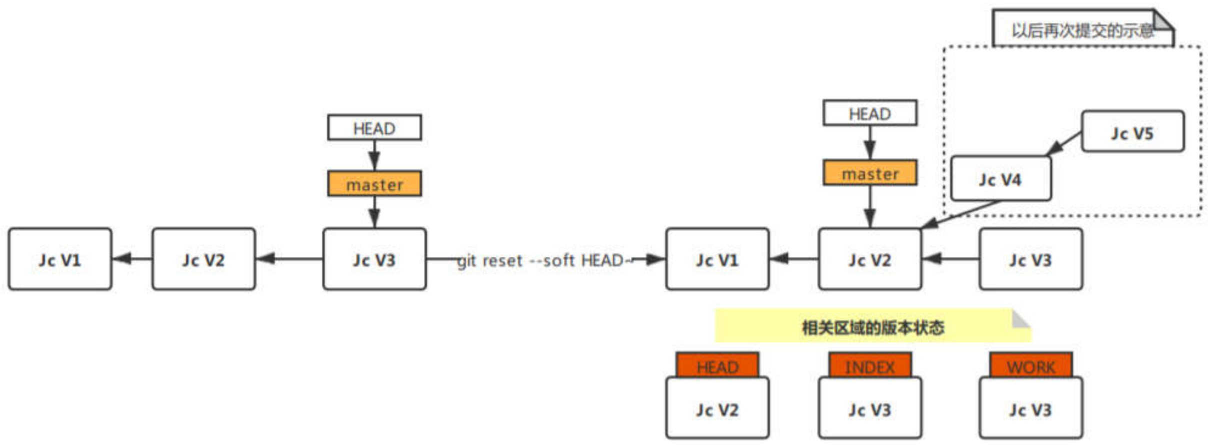
- 命令：git reset --soft HEAD~(版本库撤销原理)
  作用：移动HEAD(带着分支一起动)所指向的分支这与checkout命令是不一样的，checko只是移动HEAD不移动HEAD所执行的分支。
  直接上图吧~



```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git reset --soft HEAD~

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-files -p HEAD
git: 'cat-files' is not a git command. See 'git --help'.

The most similar command is
        cat-file

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p HEAD
tree bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
parent 4273cb3f3b0b4560c6836633c382ea71e386f675
author Jc <Jc@example.com> 1625131081 +0800
committer Jc <Jc@example.com> 1625131081 +0800

V2 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
100644 blob 0bd4328555819f5a747da844fb8f6f734ba75f9c    Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p 0bd4328555819f5a747da844fb8f6f734ba75f9c
V1
V2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ cat Jc.txt
V1
V2
V3

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git ls-files
Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git ls-files -s
100644 49de0503a08f46662347a00e328dee366958cd88 0       Jc.txt
```

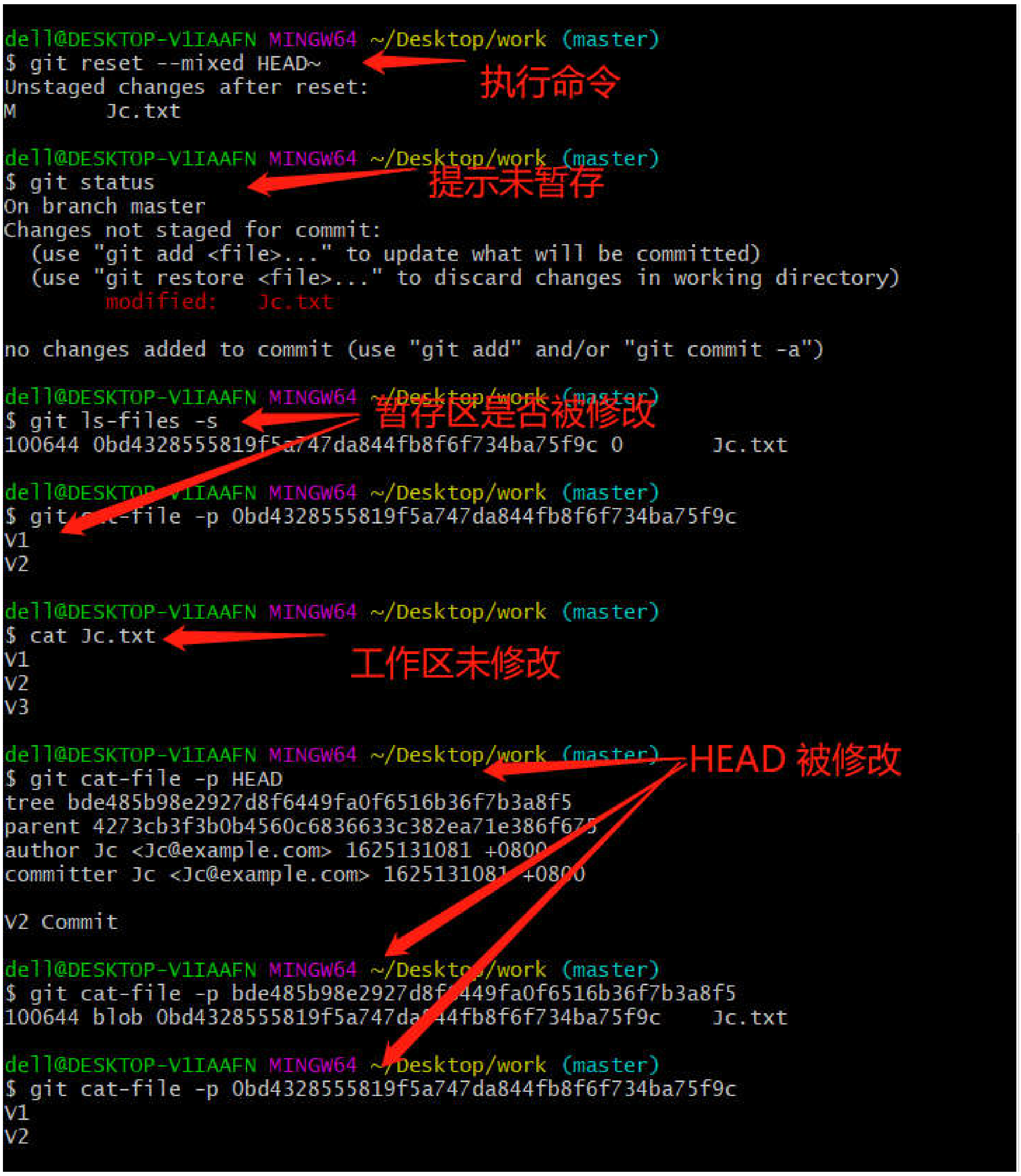


```
$ git cat-file -p 49de0503a08f46662347a00e328dee366958cd88
V1
V2
V3
```



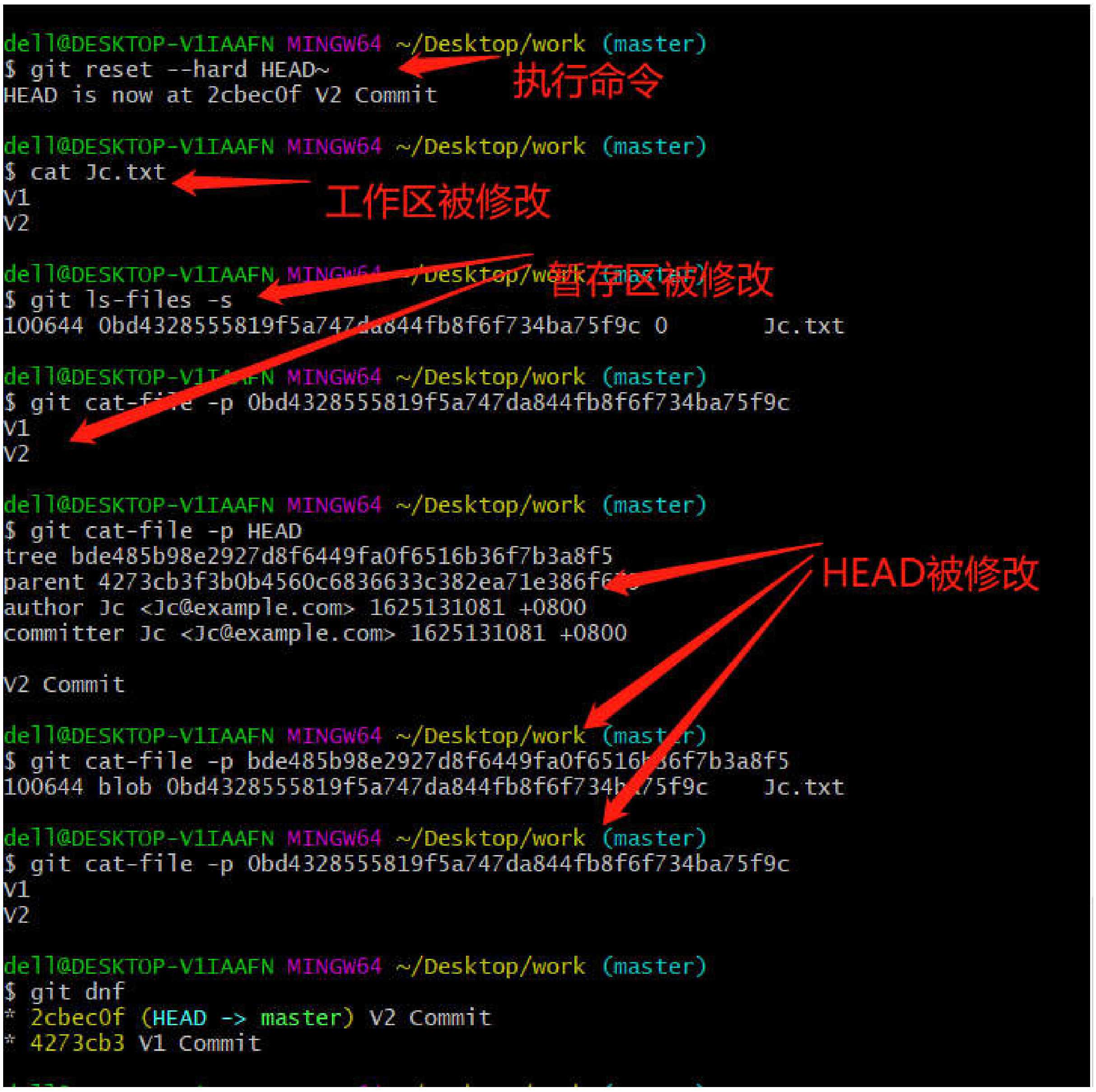
相信大家看完这图解就会明白啦~一定要记得自己尝试哦！！

- 命令：git reset --mixed HEAD~(暂存区撤销原理)
  作用：移动了HEAD(带着分支一起动)和暂存区，大家可以安照我上面的演示的方法去自行验证~我就直接上图了！



```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git reset --mixed HEAD~
Unstaged changes after reset:
M       Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jc.txt

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

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git ls-files -s
100644 0bd4328555819f5a747da844fb8f6f734ba75f9c 0       Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p 0bd4328555819f5a747da844fb8f6f734ba75f9c
V1
V2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ cat Jc.txt
V1
V2
V3

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p HEAD
tree bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
parent 4273cb3f3b0b4560c6836633c382ea71e386f675
author Jc <Jc@example.com> 1625131081 +0800
committer Jc <Jc@example.com> 1625131081 +0800

V2 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
100644 blob 0bd4328555819f5a747da844fb8f6f734ba75f9c    Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p 0bd4328555819f5a747da844fb8f6f734ba75f9c
V1
V2
```

- 命令：git reset --hard HEAD~(checkout branchName 分支切换原理。注意：checkout 只动HEAD不动分支，对工作目录是安全的。) 作用：移动了HEAD(带着分支一起动)、暂存区以及工作区，大家可以安照我上面的演示的方法去自行验证~我就直接上图了！



```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git reset --hard HEAD~
HEAD is now at 2cbec0f V2 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ cat Jc.txt
V1
V2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git ls-files -s
100644 0bd4328555819f5a747da844fb8f6f734ba75f9c 0       Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p 0bd4328555819f5a747da844fb8f6f734ba75f9c
V1
V2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p HEAD
tree bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
parent 4273cb3f3b0b4560c6836633c382ea71e386f6
author Jc <Jc@example.com> 1625131081 +0800
committer Jc <Jc@example.com> 1625131081 +0800

V2 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p bde485b98e2927d8f6449fa0f6516b36f7b3a8f5
100644 blob 0bd4328555819f5a747da844fb8f6f734ba75f9c    Jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git cat-file -p 0bd4328555819f5a747da844fb8f6f734ba75f9c
V1
V2

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 2cbec0f (HEAD -> master) V2 Commit
* 4273cb3 V1 Commit
```

四、 结尾

至此呢Git的撤退功能讲解基本完毕，如果感觉有用可以点个赞的哦！我会持续更新，如果有错误还请指出来,感谢观众老爷的赏脸。

若想获得上述内容的PDF版本移步到GitHub下载。

地址: Git 学习笔记专区。