链滴

Git 恢复进度

作者：shiweichn
原文链接：https://ld246.com/article/1533450160408
来源网站：链滴

Git恢复进度

---

1 Git stash 命令

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status -sb
## master
[root@izwz9f5nsv33jsmjcx0qw1z demo]# echo "甲乙丙" >> demo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status -sb
## master
 M demo.txt
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash
Saved working directory and index state WIP on master: 5a6b4a4 Merge commit '7b421db'
HEAD 现在位于 5a6b4a4 Merge commit '7b421db'
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash list
stash@{0}: WIP on master: 5a6b4a4 Merge commit '7b421db'
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash pop
# 位于分支 master
# 尚未暂存以备提交的变更:
#    (使用 "git add <file>..." 更新要提交的内容)
#    (使用 "git checkout -- <file>..." 丢弃工作区的改动)
#
#    修改:       demo.txt
#
修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
丢弃了 refs/stash@{0} (97554d28be5beb31be8cf470ccf3906c75f135a4)
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

git stash 可以用于保存和恢复工作进度，使用如下：

- git stash 保存当前的工作进度，会分别对工作区和暂存区的状态进行保存。

- git stash list 显示进度列表，此命令暗示了git stash 可以多次保存进度，并且在恢复的时候进行选。
- git stash pop [--index] [<stash>]
  - 若不使用任何参数,会恢复最近保存的工作进度，并将恢复的工作进度从存储的工作进度列表中除。
  - 若提供 <stash> 参数(来自于git stash list 显示的列表)，则从该<stash>中恢复，最后会从进列表中删除。
- --index 表示除了恢复工作区之外，还尝试恢复暂存区。
- git stash [save [--patch] [-k|--[no-]keep-index] [-q|--quiet] [-u|--include-untracked] [-a|--all] <message>]]
  - 这个命令其实是 git stash的完整版，即如果需要在保存工作进度的时候使用指定说明，必须用以下格式：

```
git stash save "message..."
```

  - 使用参数 --patch 会显示工作区和HEAD 之间的差异，通过对比差异文件可以决定在进度中最要保存的工作区的内容，通过编辑差异文件可以在进度中排除无关内容。

原文链接：Git 恢复进度

- 使用 -k 或 --keep-index 参数，在保存进度后不会将暂存区重置，默认会将暂存区和工作区强制置。
- git stash apply [--index] [<stash>] 除了不删除恢复的进度之外，其余的和git stash pop命令一。
- git stash drop [<stash>] 删除一个工作进度，默认删除最新的进度。
- git stash clear 删除所有存储的工作进度。
- git stash branch <branchname> <stash> 基于进度创建分支。

---

2 探秘 git stash

当执行git stash 命令时，Git实际调用了一个脚本文件实现相关功能，git --exec-path 可以看脚本的置。

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git --exec-path
/usr/libexec/git-core
[root@izwz9f5nsv33jsmjcx0qw1z demo]# ll /usr/libexec/git-core/
total 173220
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-add
-rwxr-xr-x   1 root root   36655 Mar 23  2016 git-add--interactive
-rwxr-xr-x   1 root root   22361 Mar 23  2016 git-am
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-annotate
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-apply
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-archive
-rwxr-xr-x   1 root root   11993 Mar 23  2016 git-bisect
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-bisect--helper
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-blame
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-branch
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-bundle
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-cat-file
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-check-attr
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-check-ignore
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-checkout
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-checkout-index
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-check-ref-format
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-cherry
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-cherry-pick
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-clean
-rwxr-xr-x 112 root root 1514952 Mar 23  2016 git-clone
......
```

在工作区创建/改动两个文件，然后将其中一个add到暂存区。

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status -sb
## master
M  demo.txt
?? welcome.log
```

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash
Saved working directory and index state WIP on master: 5a6b4a4 Merge commit '7b421db'
HEAD is now at 5a6b4a4 Merge commit '7b421db'
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

然后执行git stash 保存工作进度， 之后发现工作区恢复到了创建/改动这两个文件之前的状态，实际这里是使用了git reset --hard HEAD命令。

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# echo "test" >> welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git status -s
?? welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# cat welcome.log
test
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash
No local changes to save
[root@izwz9f5nsv33jsmjcx0qw1z demo]#
```

进度保存失败！由上面可以看出，本地没有被版本控制系统跟踪的文件不能保存进度，因此本地文件须先执行添加操作，再保存进度。如下：

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git add welcome.log
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash
Saved working directory and index state WIP on master: 5a6b4a4 Merge commit '7b421db'
HEAD is now at 5a6b4a4 Merge commit '7b421db'
```

3 结论

- 在用 git stash命令保存进度时，提供说明则更容易找到保存的进度。
- 每个进度的标识都是 stash@{<n>} 格式，实际上，git stash 就是用引用和引用变更日志(reflog)实现的。
    - 引用： .git/refs/stash 。引用日志：.git/logs/refs/stash
    - 对照 git stash list 和 git reflog show refs/stash的结果，可以肯定用git stash保存进度，实上会将进度保存在引用refs/stash 所指向的提交中。多次保存进度，refs/stash 中都会记录最新那次存操作的保存ID

引用 refs/stash 是如何同时保存暂存区和工作区的进度?

```
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git stash list
stash@{0}: WIP on master: 5a6b4a4 Merge commit '7b421db'
stash@{1}: WIP on master: 5a6b4a4 Merge commit '7b421db'
[root@izwz9f5nsv33jsmjcx0qw1z demo]# git log --graph --pretty=raw refs/stash -2
*   commit 1ae763017933195f64924957668317e7320ed000
|\  tree 8ebb8f1bed60121450239ab0231a971a4609f069
| | parent 5a6b4a4493b798c090aa7b314ab3359e4562b2b1
| | parent 33f7aaa95f5124fad71f5f6a00285368ed5875c9
| | author shiweichn <shiweichn@163.com> 1505907553 +0800
| | committer shiweichn <shiweichn@163.com> 1505907553 +0800
| |
| |     WIP on master: 5a6b4a4 Merge commit '7b421db'
```

```
| |
| * commit 33f7aaa95f5124fad71f5f6a00285368ed5875c9
|/  tree 8ebb8f1bed60121450239ab0231a971a4609f069
|   parent 5a6b4a4493b798c090aa7b314ab3359e4562b2b1
|   author shiweichn <shiweichn@163.com> 1505907553 +0800
|   committer shiweichn <shiweichn@163.com> 1505907553 +0800
|
|       index on master: 5a6b4a4 Merge commit '7b421db'
```

从提交历史中可以看出最新的进度保存提交是一个合并提价。WIP(Work In Progess) 代表了工区进度。而最新提交的第二个父提交，描述中包含 index on master 字样，说明这个提交代表着暂存的进度。

可以发现，这两个提交对应着同一棵树，这是因为最后一次做进度保存时工作区相对于暂存区没改变。