`git checkout` 命令用于切换分支或恢复工作树文件。`git checkout` 是 git 最常用的命令之一，同时也是一个很危险的命令，因为这条命令会重写工作区。

使用语法

```
git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]
git checkout [-q] [-f] [-m] [--detach] <commit>
git checkout [-q] [-f] [-m] [[-b|-B|--orphan] <new_branch>] [<start_point>]
git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--] <paths>…
git checkout [-p|--patch] [<tree-ish>] [--] [<paths>…]
```

描述

更新工作树中的文件以匹配索引或指定树中的版本。如果没有给出路径 - `git checkout` 还会更新 `HEAD`，将指定的分支设置为当前分支。

示例

以下是一些示例 -

示例-1

以下顺序检查主分支，将 `Makefile` 还原为两个修订版本，错误地删除 `hello.c`，并从索引中取回。

```
$ git checkout master             #(1)
$ git checkout master~2 Makefile  #(2)
$ rm -f hello.c
$ git checkout hello.c            #(3)
```

(1) 切换分支
(2) 从另一个提交中取出文件
(3)从索引中恢复 `hello.c`

如果想要检出索引中的所有 `C` 源文件，可以使用以下命令 -

```
$ git checkout -- '*.c'
```

注意:`*.c` 是使用引号的。 文件 `hello.c` 也将被检出，即使它不再在工作树中，因为文件 `globbing` 用于匹配索引中的条目(而不是在 shell 的工作树中)。

如果有一个分支也命名为 *hello.c*，这一步将被混淆为一个指令切换到该分支。应该写：

```
$ git checkout -- hello.c
```

示例-2

在错误的分支工作后，想切换到正确的分支，则使用：

```
$ git checkout mytopic
```

但是，您的“错误”分支和正确的“`mytopic`”分支可能会在在本地修改的文件中有所不同，在这种情况下，上述检出将会失败：

```
$ git checkout mytopic
error: You have local changes to 'frotz'; not switching branches.
```

可以将`-m`标志赋给命令，这将尝试三路合并：

```
$ git checkout -m mytopic
Auto-merging frotz
```

在这种三路合并之后，本地的修改没有在索引文件中注册，所以`git diff`会显示从新分支的提示之后所做的更改。

示例-3

当使用`-m`选项切换分支时发生合并冲突时，会看到如下所示：

```
$ git checkout -m mytopic
Auto-merging frotz
ERROR: Merge conflict in frotz
fatal: merge program failed
```

此时，`git diff`会显示上一个示例中干净合并的更改以及冲突文件中的更改。 编辑并解决冲突，并用常规方式用`git add`来标记它：

```
$ edit frotz # 编辑 frotz 文件中内容，然后重新添加
$ git add frotz
```

其它示例

`git checkout`的主要功能就是迁出一个分支的特定版本。默认是迁出分支的 HEAD 版本一此用法示例：

```
$ git checkout master     #//取出 master 版本的 head。
$ git checkout tag_name    #//在当前分支上 取出 tag_name 的版本
$ git checkout  master file_name  #//放弃当前对文件 file_name 的修改
$ git checkout  commit_id file_name  #//取文件 file_name 的 在 commit_id 是的版本。commit_id 为 git commit 时的 sha 值。


$ git checkout -b dev/1.5.4 origin/dev/1.5.4


# 从远程 dev/1.5.4 分支取得到本地分支/dev/1.5.4
$ git checkout -- hello.rb
#这条命令把 hello.rb 从 HEAD 中签出.
$ git checkout .
```

```
#这条命令把 当前目录所有修改的文件 从 HEAD 中签出并且把它恢复成未修改时的样子.
#注意：在使用 git checkout 时，如果其对应的文件被修改过，那么该修改会被覆盖掉。
```