

# 2022-2023年春季学期 北大工学院《并行程序设计》

## 11. Git版本管理


授课教员：陈默涵





## 1. Git背景介绍

## 2. Git命令介绍

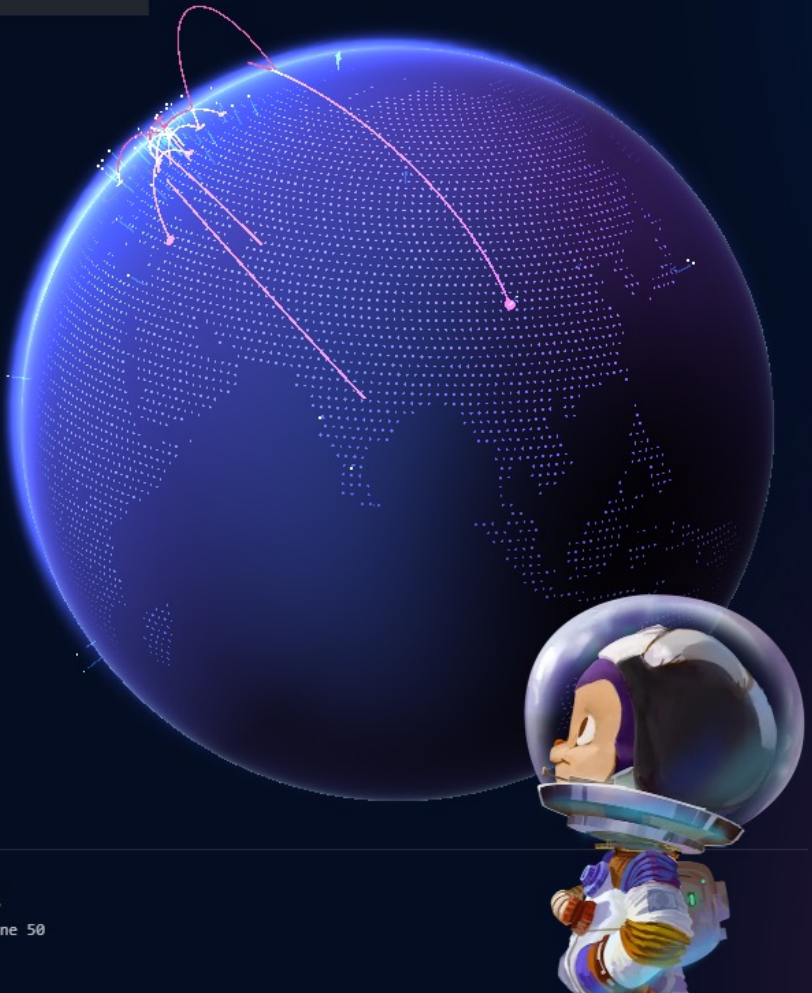
 [Why GitHub?](#) [Team](#) [Enterprise](#) [Explore](#) [Marketplace](#) [Pricing](#)  [Sign in](#) [Sign up](#)

按 F11 即可退出全屏模式

# Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

[Sign up for GitHub](#)



**56+ million**  
Developers

**3+ million**  
Organizations

**100+ million**  
Repositories

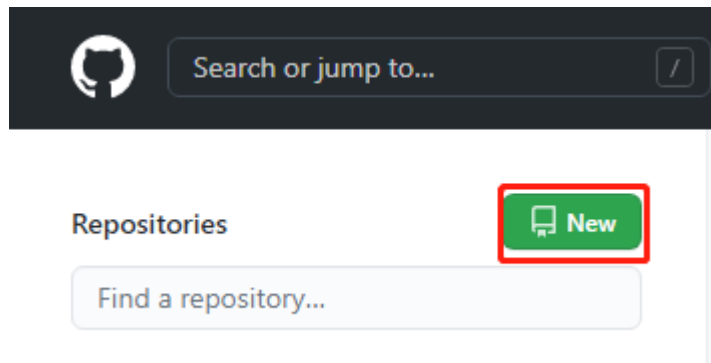
**72%**  
Fortune 50

- Gitee是开源中国（OSChina）推出的基于Git的代码托管服务
- Gitee包括三个版本，分别是：社区版、企业版和高校版
- 可以通过 SSH 或者 HTTP 的方式提交和管理代码，也可以通过 Web 的方式在线阅读、编辑代码。
- 通过 Fork 和 Pull Request 功能您可以方便的向其它项目贡献代码，你也可以为项目创建 Issue 和 Wiki

网址：<https://gitee.com/>

- 注册登录之后可以打开仓库，建立新仓库，并通过Linux终端进行代码的交互

# 回到Github: 创建仓库



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

 YuLiu98 ▾

Repository name \*

MDtest ✓

Great repository names are short and memorable. Need inspiration? How about [studious-meme?](#)

Description (optional)

give an example

☐



Public

Anyone on the internet can see this repository. You choose who can commit.

☒



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☒

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)


.gitignore template: C++ ▾

☒

Choose a license


A license tells others what they can and can't do with your code. [Learn more.](#)


License: GNU General Public ... ▾


This will set  main as the default branch. Change the default name in your [settings](#).


Create repository


# 创建仓库


 YuLiu98 / MDtest Private


 Unwatch


 Code


 Issues


 Pull requests


 Actions


 Projects


 Security

 Insights

 Settings

 main ▾


 1 branch


 0 tags




Go to file

Add file ▾


Code ▾

 YuLiu98 Initial commit

89a3d4f now  1 commit

|  |                |     |
|--|----------------|-----|
|  .gitignore | Initial commit | now |
|  LICENSE    | Initial commit | now |
|  README.md  | Initial commit | now |

README.md

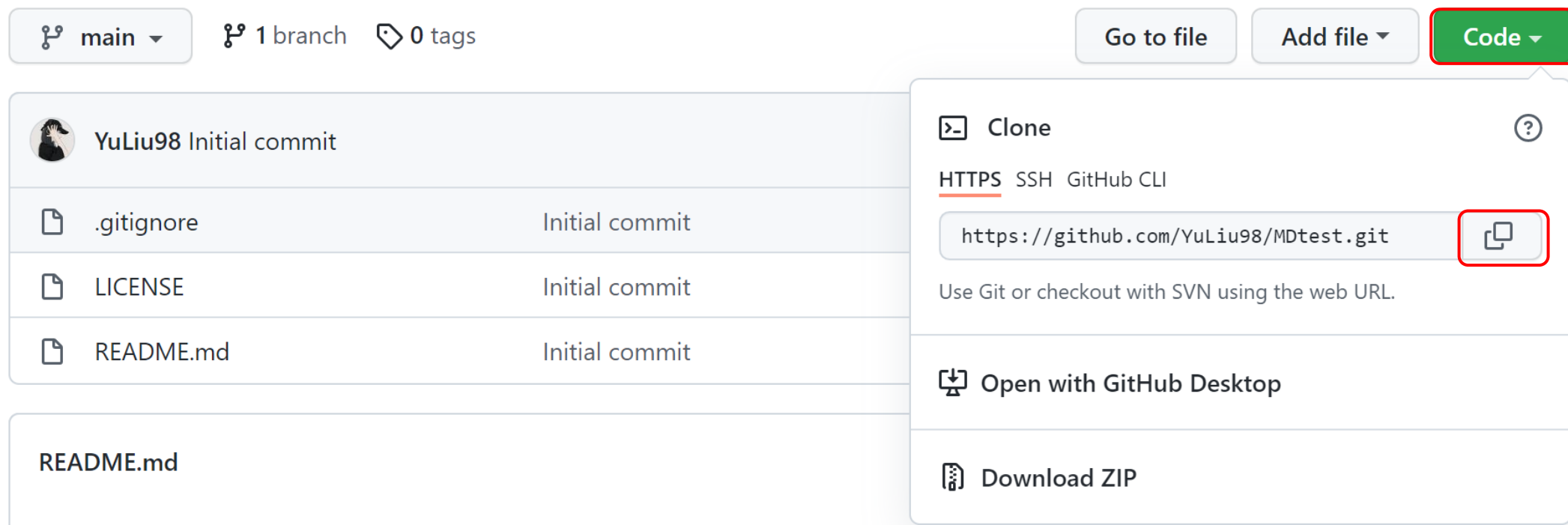


## MDtest

---

give an example

# 克隆远程仓库到本地



git clone <https://github.com/YuLiu98/MDtest.git>

```
(base) liuyu@Rainfall:~/MDtest$ ls -la
.  ..  .git  .gitignore  LICENSE  README.md
```

此时会得到一个MDtest文件夹，这不是普通的文件夹，进入，输入ls -la，会发现有.git隐藏文件夹，里面就会有一些用户信息，commit信息等

# 配置用户信息

```
git config --global user.name "zhangsan"
```

```
git config --global user.email zhangsan@pku.edu.cn
```

```
git config --global user.password
```

(--global表示全局配置，一般一个账号就用全局配置)

其中 **email**决定**github**的账号， **user.name**只在没有用该邮箱注册**github**时起作用





## 1. Git背景介绍

## 2. Git命令介绍

# 本地仓库新增文件

进入MDtest文件夹，创建一些文件，比如main.cpp, Makefile, fun.cpp

`git add .`

将当前文件夹所有文件添加至索引库，这样就能跟踪这些文件了

`git status` (显示当前状态)

```
位于分支 master
初始提交
要提交的变更:
  (使用 "git rm --cached <文件>..." 以取消暂存)

    新文件:   Makefile
    新文件:   fun.cpp
    新文件:   main.cpp
```

如果发现文件不对，可以进行一些修改，重新`git add`。

# 提交到远程仓库

git commit -m "First commit "

```
[master (根提交) 9c6d98c] First commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Makefile
create mode 100644 fun.cpp
create mode 100644 main.cpp
```

git push 推到远程仓库

```
对象计数中: 3, 完成.
Delta compression using up to 32 threads.
压缩对象中: 100% (2/2), 完成.
写入对象中: 100% (3/3), 224 bytes | 0 bytes/s, 完成.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Qianruipku/qianrui-MD.git
 * [new branch]      master -> master
```

git status

```
位于分支 master
您的分支与上游分支 'origin/master' 一致。
无文件要提交，干净的工作区
```

# 提交到远程仓库

`git push <远程主机名> <本地分支名>:<远程分支名>`

之前运行的git push 省略了以上三个名字：

查看默认的远程主机名：`git remote`，一般默认为origin

查看默认的本地分支名：`git branch`，一般为master

远程分支名：一般与本地分支名相同

查看所有远程分支：`git branch -r`

所以前面的git push实际是运行了：

`git push origin master : master`

# 从远程仓库获取最新版本

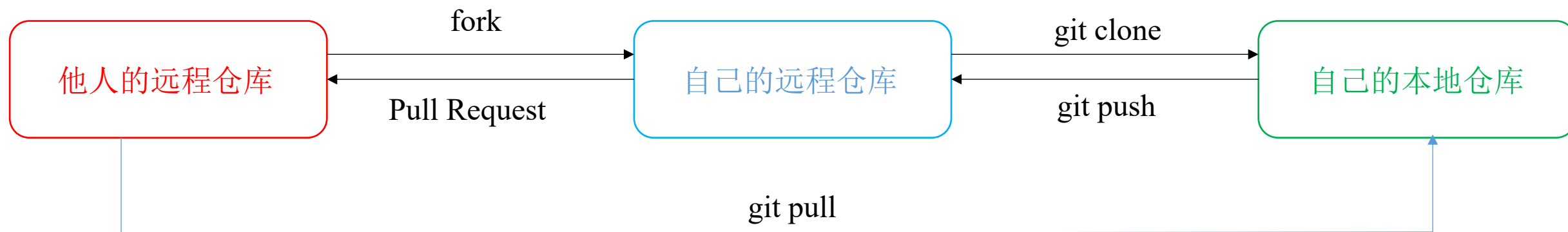
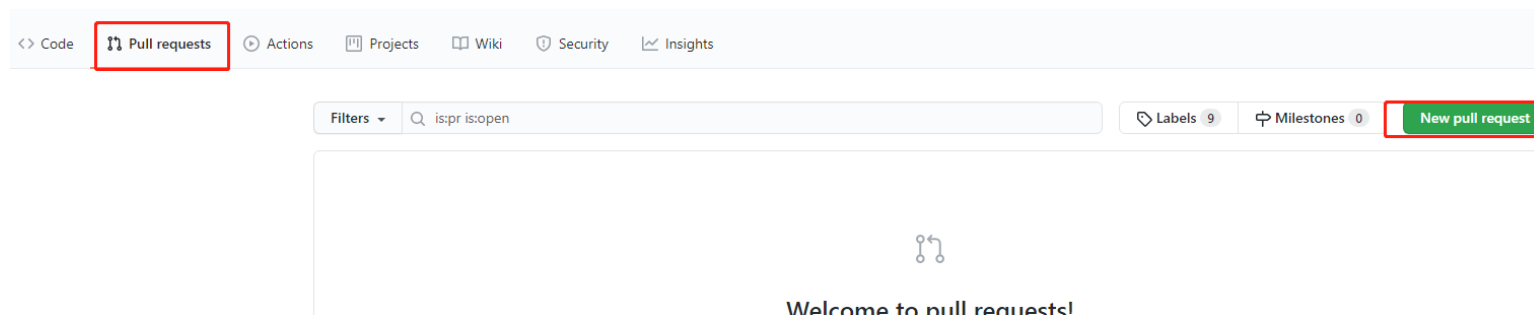
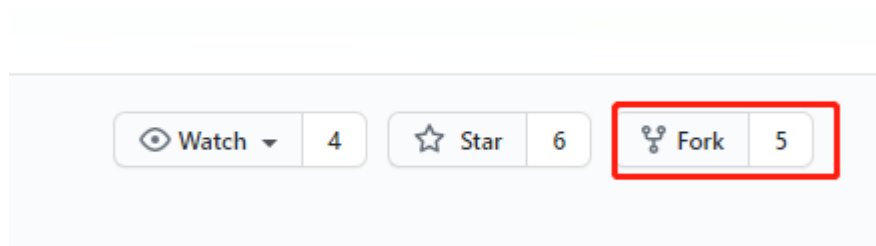
```
git pull <远程主机名> <远程分支名>:<本地分支名>
```

多人开发时，先git pull拉取远程仓库中的其他人的修改，合并到本地仓库之后再git push

如果本地仓库和远程仓库产生冲突，则需要手动修改

# Pull Request (PR)：提交代码

多人开发时，流程更为复杂



# Git log: 提交历史记录

`git log` 显示提交历史

`git log --graph` 显示提交历史并且分支合并信息

`git log --author=xxx` 显示某个作者所有提交信息

```
commit be7707fff27e201ae0d413793f3b2a6d45358d06 (HEAD -> reconstruction, origin/reconstruction)
```

```
Merge: 1c418e67 1699f96f
```

```
Author: dyzheng <49852742+dyzheng@users.noreply.github.com>
```

```
Date:   Fri Oct 15 15:55:10 2021 +0800
```

```
Merge pull request #396 from deepmodeling/develop
```

```
Update branch reconstruction
```

```
commit 1699f96fbaac9bc2d3de7916697b53b240e422cf
```

```
Merge: 48253c0f 071fc8e3
```

```
Author: dyzheng <49852742+dyzheng@users.noreply.github.com>
```

```
Date:   Fri Oct 15 15:52:43 2021 +0800
```

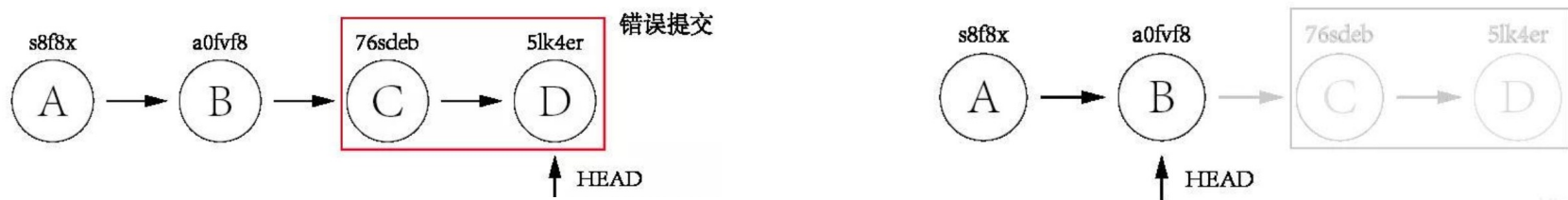
```
Merge pull request #395 from deepmodeling/newelpa
```

```
Newelpa and reconstruction merge to develop branch, compatible for current deepks branch.
```

# Git reset

假设现在系统有两次正常提交（A和B），而C和D是错误提交。现在要把C和D去掉，而此时HEAD指针（当前最新版本）已经指向D。为此，我们要把HEAD指针移动到B。

运行命令：`git reset <commit-num-B>`（commit编号可以查看git log得到）



运行之后，HEAD指针会移动到B，但是本地的代码仍处在D的状态。要想把代码也改为B的状态，需要 `git reset --hard <commit-num-B>`。

- 慎用--hard，未提交(commit)的修改不能被恢复！

```
(base) fortneu49@Eisbrecher:~/abacus-develop$ git log
commit d9c3dada732a2b96d3510bd9a2aa201665343ceb
merge: 5552742 T157e55
Author: dyzheng <zhengdy@dp.tech>
```



# Git reset

在本地执行git reset (--hard)后，远程仓库指针和代码仍然保持在D不变。若要更新远程仓库就需要强推：运行git push -f

- 慎用-f，远程仓库会失去所有C和D的信息

若要找回D的信息，可运行git reflog 找到reset操作对应的D版本号，重新git reset回去。然后可以重新git push到远程。

```
5552742 HEAD@{0}: reset: moving to HEAD^  
d9c3dad HEAD@{1}: checkout: moving from deepks to develop
```

```
• (base) fortneu49@Eisbrecher:~/abacus-develop$ git reset --hard 5552742  
HEAD 现在位于 5552742 Merge pull request #1204 from ddhss/develop
```

# 分支管理

- 何时需要多分支？——稳定分支和开发分支；多人协同，分别写各自的模块
- `git branch` 查看所有分支和当前分支
- `git checkout <branch_name>` 切换到某个分支
  - 切换之前要保证工作区没有尚未提交的修改，否则会报错。
  - 可以`git add`并且`git commit`提交所有修改 / `git stash`暂存修改 / `git reset --hard`删除修改

```
• (base) fortneu49@Eisbrecher:~/abacus-develop$ git branch
  LCAO-refactor
  TDDFT
  deepks
* develop
```

# 分支管理

- `git checkout -b <new_branch>` 从当前状态新建分支并切换过去
- `git branch -d <branch_name>` 删除分支
- `git checkout <file_name>` 还原某个文件的修改（checkout的另一个作用）
  - 和`git reset--hard`一样，不能恢复！
- `git stash` 暂存修改
  - 做完push/checkout等操作后，可以`git stash pop`还原修改（弹出暂存）

```
⊗ (base) fortneu49@Eisbrecher:~/abacus-develop$ git checkout deepks
error: Your local changes to the following files would be overwritten by checkout:
    source/module_deepks/LCAO_deepks_fdelta.cpp
    source/module_deepks/LCAO_deepks_pdm.cpp
Please, commit your changes or stash them before you can switch branches.
Aborting
```

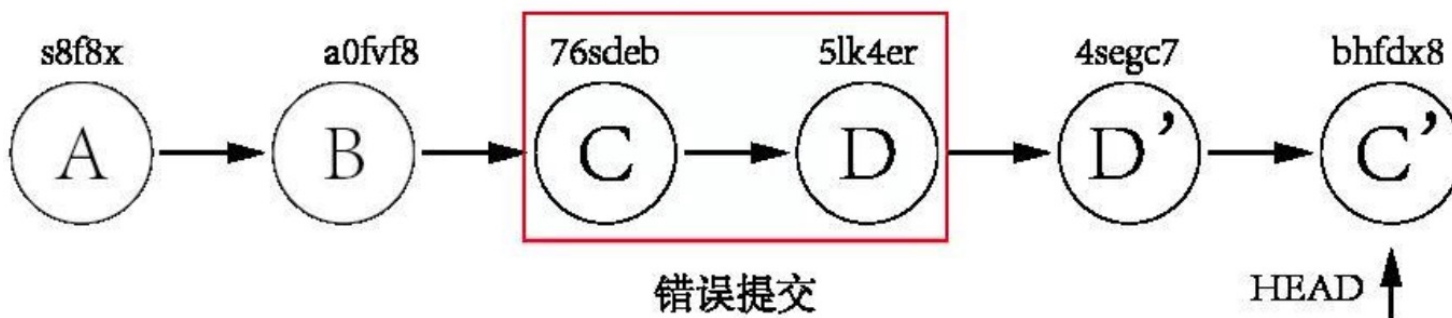
```
• (base) fortneu49@Eisbrecher:~/abacus-develop$ git stash
Saved working directory and index state WIP on develop: d9c3dad Merge pull request #1206 from deepmodeling/comment
HEAD 现在位于 d9c3dad Merge pull request #1206 from deepmodeling/comment
• (base) fortneu49@Eisbrecher:~/abacus-develop$ git checkout deepks
切换到分支 'deepks'
```

# 想要还原代码怎么办: Revert

首先，运行 `git revert 5lk4er`，创建D'，抹去D的影响

其次，运行 `git revert 76sdeb`，创建C'，抹去C的影响

这样操作的话，保留了C和D的信息，HEAD指针是往前移动的，因此可以直接使用 `git push` 命令推动到远程仓库里，一般是鼓励大家使用的

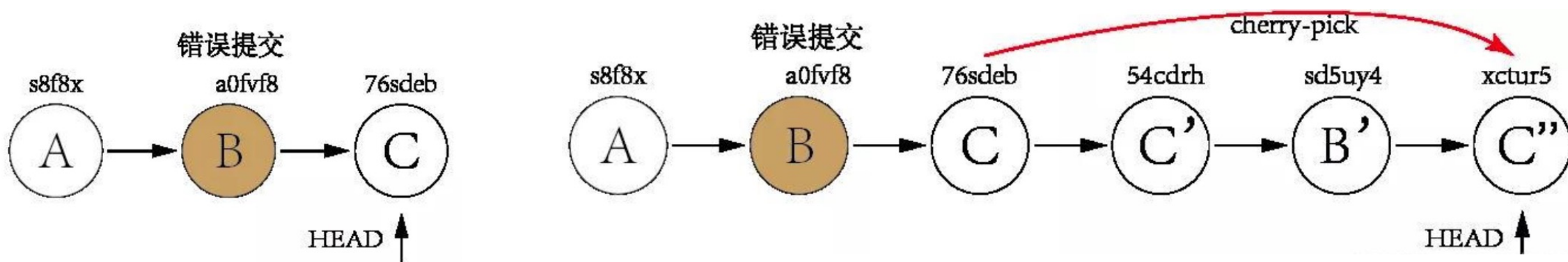


# Cherry-Pick

假如有三个提交，中间B是错误提交，但是C是正确提交，如何处理？

git revert命令先把 C 提交和B 提交全部回退，再使用 cherry-pick 命令将 C 提交重新再生成一个新的提交 C' '，这样就实现了将 B提交回退的需求。

运行命令：`git cherry-pick 76sdeb`



# Git命令小结：方便进行代码记录

进入本地文件夹MDtest，进行一些修改（此时修改在**工作区**）

**git add .** （此时修改在**缓存区**）

**git status** （查看工作区和缓存区状态）

**git commit -m "<description>"** （修改提交至**本地仓库**）

**git push origin <branch\_name>** （修改提交至**远程仓库**）

这时发现某个提交写错了（且已经被推到远程），想要恢复之前的版本：

**git log** （找到错误的commit编号）

**git reset -- hard <commit-num>** （本地仓库的修改被退回）

正准备**git push -f**，发现别人也改了远程仓库，强行push会覆盖掉别人的修改！

**git reflog** （找到reset前的commit编号）

**git reset -- hard <wrong-commit-num>** （reset回去）

**git revert <wrong-commit-num>** （用新的提交撤销之前的错误修改）

自己改到一半，发现需要基于别人的修改才能继续改下去：

**git stash** （自己的修改在本地被暂存）

**git pull origin <branch\_name>** （别人的修改被拉到本地）

**git stash pop** （在本地还原自己的修改并应用于pull下来的修改之上）

终于改完了，照例进行**add-commit-push**三步，大功告成！