**[趣谈、浅析CRLF和LF](http://www.cnblogs.com/ArsenalfanInECNU/p/6086665.html)**

<http://www.cnblogs.com/ArsenalfanInECNU/p/6086665.html>

**CR(回车)，LF(换行)**

**Dos和windows采用回车+换行CR/LF表示下一行,**  
**而UNIX/Linux采用换行符LF表示下一行，**  
**苹果机(MAC OS系统)也采用换行符LF表示下一行。**(注：有些人认为Mac Style是CR，我查了一些资料，发现应该是这样的：老的Mac是CR，后来的OSX全部改成了LF)  
**CR用符号’\r’表示, LF使用’\n’符号表示。**

**在Git中如何转换？**

在Git通过下面的命令配置

$git config --global core.autocrlf true

# Configure Git on Windows to properly handle line endings

解释：core.autocrlf是git中负责处理line endings的变量，可以设置三个值--true,inout,false.

设置成三个值会有什么效果呢？

If core.autocrlf is set to true, that means that any time you add a file to the git repo that git thinks is a text file, it will turn all CRLF line endings to just LF before it stores it in the commit.。

设置为true，添加文件到git仓库（即提交代码）时，git将其视为文本文件。他将把crlf变成lf。

If core.autocrlf is set to false, no line-ending conversion is ever performed, so text files are checked in as-is. This usually works ok。

设置为false时，line-endings将不做转换操作。文本文件保持原来的样子。

设置为input时，添加文件git仓库（即提交代码）时，git把crlf编成lf。当有人Check代码（即从git仓库迁出代码）时，按照input的方式，即不做修改，完全照搬git仓库里的。因此在window操作系统下，尽量不要使用这个设置。

**Yet another way to show how autocrlf works**

1) true: x -> LF -> CRLF

2) input: x -> LF -> LF

3) false: x -> x -> x

大神, 我发现你少说了一个非常重要的问题, 一般在添加完公钥之后, 很多人经常会遇到 下面这种问题

[root@wdOS dtc2]*# git clone git@github.com:xxxxx/dtc2*

Cloning into 'dtc2'...

The authenticity of host 'github.com (192.30.252.128)' can't be established.

RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.

Are you sure you want to continue connecting (yes/no)?

\*\*Host key verification failed.\*\*

fatal: Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.

说是主机密钥验证失败, 后来google了老半天时间发现是缺少了 known\_hosts 文件, 而且必须生成 github.com 的ip执行内容. 如下:

[root@iZ94xpmg9boZ .ssh]*# ssh git@github.com*

The authenticity of host 'github.com (192.30.252.131)' can't be established.

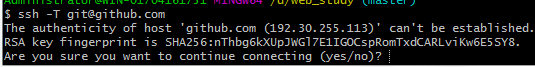
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.

Are you sure you want to continue connecting (yes/no)? yes #这里写 yes

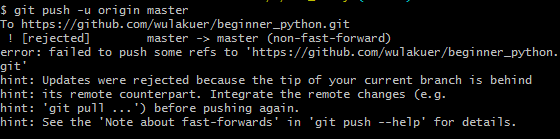
Warning: Permanently added 'github.com,192.30.252.131' (RSA) to the list of known hosts.

Permission denied (publickey).

oh my god , 文中 SSH警告 部分已经说了, 我没看清,多此一举 ... 傻了我



[Updates were rejected because the tip of your current branch is behind](http://blog.csdn.net/huangbin10025/article/details/51371715)



有如下几种解决方法：  
1.使用强制push的方法：  
$ git push -u origin master -f   
这样会使远程修改丢失，一般是不可取的，尤其是多人协作开发的时候。  
2.push前先将远程repository修改pull下来  
$ git pull origin maste  
$ git push -u origin master  
3.若不想merge远程和本地修改，可以先创建新的分支：  
$ git branch [name]  
然后push  
$ git push -u origin [name]

[git无法pull仓库refusing to merge unrelated histories](http://blog.csdn.net/lindexi_gd/article/details/52554159)



本文讲的是把[**Git**](http://lib.csdn.net/base/git)在最新2.9.2，合并pull两个不同的项目，出现的问题如何去解决fatal: refusing to merge unrelated histories

我在Github新建一个仓库，写了License，然后把本地一个写了很久仓库上传。

先pull，因为两个仓库不同，发现refusing to merge unrelated histories，无法pull

因为他们是两个不同的项目，要把两个不同的项目合并，**[git](http://lib.csdn.net/base/git" \t "_blank" \o "Git知识库)**需要添加一句代码，在git pull，这句代码是在git 2.9.2版本发生的，最新的版本需要添加--allow-unrelated-histories

假如我们的源是origin，分支是master，那么我们 需要这样写git pull origin master --allow-unrelated-histories需要知道，我们的源可以是本地的路径

Git 命令行

Git 远程仓库命令github

**git remote add origin** [**https://github.com/wulakuer/beginner\_python.git**](https://github.com/wulakuer/beginner_python.git)

alias=origin

url=https://

**ssh-keygen -t rsa -C "youremail@example.com"**

**ssh -T git@github.com**

**git pull origin master --allow-unrelated-histories**

=git fetch + git merge

**git push -u origin master**

**git fetch**

**git remote**

与本地仓库中使用git branch相似

**git remote -v**

**git remote rm [别名]**

**$ git clone git://github.com/schacon/grit.git**

本地仓库使用git XXX，远程仓库使用git remote XXX(除了pull,push,fetch)

**Git基本操作**

**git init**

**git clone url alias**

**git add**

**git status -s**

**git diff --cached –stat HEAD**

**git commit -m -a -am**

**git reset HEAD**

**git rm --cached**

**git mv**

git 分支管理

git branch branchname

git checkout

git merge

git branch -b branchname

git banch -d branchname

**Git查看提交历史**

**git log –oneline –graph –author=wulakuer**

**Git标签**

**Git tag -a v1.0 -m “comments” SHA码（追加tag时）**

Git ignore

<http://blog.csdn.net/benkaoya/article/details/7932370>

$ git check-ignore -v App.**class**

**.gitignore:3:\*.class App.class**

**Git忽略规则及.gitignore规则不生效的解决办法**

在git中如果想忽略掉某个文件，不让这个文件提交到版本库中，可以使用修改根目录中 .gitignore 文件的方法（如无，则需自己手工建立此文件）。这个文件每一行保存了一个匹配的规则例如：

[?](http://www.pfeng.org/archives/840)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # 此为注释 – 将被 Git 忽略    \*.a       # 忽略所有 .a 结尾的文件  !lib.a    # 但 lib.a 除外  /TODO     # 仅仅忽略项目根目录下的 TODO 文件，不包括 subdir/TODO  build/    # 忽略 build/ 目录下的所有文件  doc/\*.txt # 会忽略 doc/notes.txt 但不包括 doc/server/arch.txt |

规则很简单，不做过多解释，但是有时候在项目开发过程中，突然心血来潮想把某些目录或文件加入忽略规则，按照上述方法定义后发现并未生效，原因是.gitignore只能忽略那些原来没有被track的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。那么解决方法就是先把本地缓存删除（改变成未track状态），然后再提交：

[?](http://www.pfeng.org/archives/840)

|  |  |
| --- | --- |
| 1  2  3 | git rm -r --cached .  git add .  git commit -m 'update .gitignore' |