## 关于Git的礼节

(这里的内容本来是《<u>怎样尊重一个程序员</u>》的一小节,但由于Git的使用引起了很普遍的不尊重程序员的现象,现在特别将这一节提出来单独成文。)

Git是现在最流行的代码版本控制工具。用外行话说,Git就是一个代码的"仓库"或者"保管",这样很多人修改了代码之后,可以知道是谁改了哪一块。其实不管什么工具,不管是编辑器,程序语言,还是版本控制工具,比起程序员的核心思想来,都是次要的东西,都是起辅助作用的。可是Git这工具似乎特别惹人恼火。

Git并不像很多人吹嘘的那么好用,其中有明显的蹩脚设计。跟Unix的传统一脉相承,Git没有一个良好的包装,设计者把自己的内部实现细节无情地泄露给了用户,让用户需要琢磨者设计者内部到底怎么实现的,否则很多时候不知道该怎么办。用户被迫需要记住挺多稀奇古怪的命令,而且命令行的设计也不怎么合理,有时候你需要加-f之类的参数,各个参数的位置可能不一致,而且加了还不一定能起到你期望的效果。各种奇怪的现象,比如"head detached",都强迫用户去了解它内部是怎么设计的。随着Git版本的更新,新的功能和命令不断地增加,后来你终于看到命令行里出现了foreach,才发现它的命令行就快变成一个(劣质的)程序语言。如果你了解ydiff的设计思想,就会发现Git之类基于文本的版本控制工具,其实属于古代的东西。然而很多人把Git奉为神圣,就因为它是Linus Torvalds设计的。

Git最让人恼火的地方并不是它用起来麻烦,而是它的"资深用户"们居高临下的态度给你造成的心理阴影。好些人因为自己"精通Git"就以为高人一等,摆出一副专家的态度。随着用户的增加,Git最初的设计越来越被发现不够用,所以一些约定俗成的规则似乎越来越多,可以写成一本书!跟Unix的传统一脉相承,Git给你很多可以把自己套牢的"机制",到时候出了问题就怪你自己不知道。所以你就经常听有人煞有介事的说:"并不是Git允许你这么做,你就可以这么做的!Unix的哲学是不阻止傻人做傻事……"如果你提交代码时不知道Git用户一些约定俗成的规则,就会有人嚷嚷:"rebase了再提交!""不要push到master!""不要merge!""squash commits!"如果你不会用git submodule之类的东西,有人可能还会鄙视你,说:"你应该知道这些!"

打个比方,这样的嚷嚷给人的感觉是,你得了奥运会金牌之后,把练习用的器材还回到器材保管科,结果管理员对你大吼:"这个放这边!那个放那边!懂不懂规矩啊你?"看出来问题了吗?程序员提交了有高价值的代码(奥运金牌),结果被一些自认为Git用的很熟的人(器材保管员)厉声呵斥。

一个尊重程序员的公司文化,就应该把程序员作为运动健将,把程序员的代码放在尊贵的地位。其它的工具,都应该像器材保管科一样。我们尊重这些器材保管员,然而如果运动员们不懂你制定的器材摆放规矩,也应该表示出尊重和理解,说话应该和气有礼貌,不应该骑到他们头上。所以,对于Git的一些命令和用法,我建议大家向新手介绍时,这样开场:"你本来不该知道这些的,可是现在我们没有更好的工具,所以得这样弄一下……"