Git挂钩

和其他版本控制系统一样，当某些重要事件发生时，Git 以调用自定义脚本。有两组挂钩：客户端和服务器端。客户端挂钩用于客户端的操作，如提交和合并。服务器端挂钩用于 Git 服务器端的操作，如接收被推送的提交。你可以随意地使用这些挂钩，下面会讲解其中一些。

安装一个挂钩

挂钩都被存储在 Git 目录下的hooks子目录中，即大部分项目中的.git/hooks。 Git 默认会放置一些脚本样本在这个目录中，除了可以作为挂钩使用，这些样本本身是可以独立使用的。所有的样本都是shell脚本，其中一些还包含了Perl的脚本，不过，任何正确命名的可执行脚本都可以正常使用 — 可以用Ruby或Python，或其他。在Git 1.6版本之后，这些样本名都是以.sample结尾，因此，你必须重新命名。在Git 1.6版本之前，这些样本名都是正确的，但这些样本不是可执行文件。

把一个正确命名且可执行的文件放入 Git 目录下的hooks子目录中，可以激活该挂钩脚本，因此，之后他一直会被 Git 调用。随后会讲解主要的挂钩脚本。

客户端挂钩

有许多客户端挂钩，以下把他们分为：提交工作流挂钩、电子邮件工作流挂钩及其他客户端挂钩。

提交工作流挂钩

有 4个挂钩被用来处理提交的过程。pre-commit挂钩在键入提交信息前运行，被用来检查即将提交的快照，例如，检查是否有东西被遗漏，确认测试是否运行，以及检查代码。当从该挂钩返回非零值时，Git 放弃此次提交，但可以用git commit --no-verify来忽略。该挂钩可以被用来检查代码错误（运行类似lint的程序），检查尾部空白（默认挂钩是这么做的），检查新方法（译注：程序的函数）的说明。

prepare-commit-msg挂钩在提交信息编辑器显示之前，默认信息被创建之后运行。因此，可以有机会在提交作者看到默认信息前进行编辑。该挂钩接收一些选项：拥有提交信息的文件路径，提交类型，如果是一次修订的话，提交的SHA-1校验和。该挂钩对通常的提交来说不是很有用，只在自动产生的默认提交信息的情况下有作用，如提交信息模板、合并、压缩和修订提交等。可以和提交模板配合使用，以编程的方式插入信息。

commit-msg挂钩接收一个参数，此参数是包含最近提交信息的临时文件的路径。如果该挂钩脚本以非零退出，Git 放弃提交，因此，可以用来在提交通过前验证项目状态或提交信息。本章上一小节已经展示了使用该挂钩核对提交信息是否符合特定的模式。

post-commit挂钩在整个提交过程完成后运行，他不会接收任何参数，但可以运行git log -1 HEAD来获得最后的提交信息。总之，该挂钩是作为通知之类使用的。

提交工作流的客户端挂钩脚本可以在任何工作流中使用，他们经常被用来实施某些策略，但值得注意的是，这些脚本在clone期间不会被传送。可以在服务器端实施策略来拒绝不符合某些策略的推送，但这完全取决于开发者在客户端使用这些脚本的情况。所以，这些脚本对开发者是有用的，由他们自己设置和维护，而且在任何时候都可以覆盖或修改这些脚本。

E-mail工作流挂钩

有3个可用的客户端挂钩用于e-mail工作流。当运行git am命令时，会调用他们，因此，如果你没有在工作流中用到此命令，可以跳过本节。如果你通过e-mail接收由git format-patch产生的补丁，这些挂钩也许对你有用。

首先运行的是applypatch-msg挂钩，他接收一个参数：包含被建议提交信息的临时文件名。如果该脚本非零退出，Git 放弃此补丁。可以使用这个脚本确认提交信息是否被正确格式化，或让脚本编辑信息以达到标准化。

下一个在git am运行期间调用是pre-applypatch挂钩。该挂钩不接收参数，在补丁被运用之后运行，因此，可以被用来在提交前检查快照。你能用此脚本运行测试，检查工作树。如果有些什么遗漏，或测试没通过，脚本会以非零退出，放弃此次git am的运行，补丁不会被提交。

最后在git am运行期间调用的是post-applypatch挂钩。你可以用他来通知一个小组或获取的补丁的作者，但无法阻止打补丁的过程。

其他客户端挂钩

pre-rebase挂钩在衍合前运行，脚本以非零退出可以中止衍合的过程。你可以使用这个挂钩来禁止衍合已经推送的提交对象，Git pre-rebase挂钩样本就是这么做的。该样本假定next是你定义的分支名，因此，你可能要修改样本，把next改成你定义过且稳定的分支名。

在git checkout成功运行后，post-checkout挂钩会被调用。他可以用来为你的项目环境设置合适的工作目录。例如：放入大的二进制文件、自动产生的文档或其他一切你不想纳入版本控制的文件。

最后，在merge命令成功执行后，post-merge挂钩会被调用。他可以用来在 Git 无法跟踪的工作树中恢复数据，诸如权限数据。该挂钩同样能够验证在 Git 控制之外的文件是否存在，因此，当工作树改变时，你想这些文件可以被复制。

服务器端挂钩

除了客户端挂钩，作为系统管理员，你还可以使用两个服务器端的挂钩对项目实施各种类型的策略。这些挂钩脚本可以在提交对象推送到服务器前被调用，也可以在推送到服务器后被调用。推送到服务器前调用的挂钩可以在任何时候以非零退出，拒绝推送，返回错误消息给客户端，还可以如你所愿设置足够复杂的推送策略。

pre-receive 和 post-receive

处理来自客户端的推送（push）操作时最先执行的脚本就是 pre-receive 。它从标准输入（stdin）获取被推送引用的列表；如果它退出时的返回值不是0，所有推送内容都不会被接受。利用此挂钩脚本可以实现类似保证最新的索引中不包含非fast-forward类型的这类效果；抑或检查执行推送操作的用户拥有创建，删除或者推送的权限或者他是否对将要修改的每一个文件都有访问权限。

post-receive 挂钩在整个过程完结以后运行，可以用来更新其他系统服务或者通知用户。它接受与 pre-receive相同的标准输入数据。应用实例包括给某邮件列表发信，通知实时整合数据的服务器，或者更新软件项目的问题追踪系统 —— 甚至可以通过分析提交信息来决定某个问题是否应该被开启，修改或者关闭。该脚本无法组织推送进程，不过客户端在它完成运行之前将保持连接状态；所以在用它作一些消耗时间的操作之前请三思。

update

update 脚本和 pre-receive 脚本十分类似。不同之处在于它会为推送者更新的每一个分支运行一次。假如推送者同时向多个分支推送内容，pre-receive 只运行一次，相比之下 update 则会为每一个更新的分支运行一次。它不会从标准输入读取内容，而是接受三个参数：索引的名字（分支），推送前索引指向的内容的 SHA-1 值，以及用户试图推送内容的 SHA-1 值。如果 update 脚本以退出时返回非零值，只有相应的那一个索引会被拒绝；其余的依然会得到更新。