查看提交历史

在提交了若干更新之后，又或者克隆了某个项目，想回顾下提交历史，可以使用 git log 命令查看。

接下来的例子会用我专门用于演示的 simplegit 项目，运行下面的命令获取该项目源代码：

git clone git://github.com/schacon/simplegit-progit.git

然后在此项目中运行 git log，应该会看到下面的输出：

$ git log

commit ca82a6dff817ec66f44342007202690a93763949

Author: Scott Chacon <schacon@gee-mail.com>

Date: Mon Mar 17 21:52:11 2008 -0700

changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 16:40:33 2008 -0700

removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 10:31:28 2008 -0700

first commit

默认不用任何参数的话，git log 会按提交时间列出所有的更新，最近的更新排在最上面。看到了吗，每次更新都有一个 SHA-1 校验和、作者的名字和电子邮件地址、提交时间，最后缩进一个段落显示提交说明。

git log 有许多选项可以帮助你搜寻感兴趣的提交，接下来我们介绍些最常用的。

我们常用 -p 选项展开显示每次提交的内容差异，用 -2 则仅显示最近的两次更新：

$ git log -p -2

commit ca82a6dff817ec66f44342007202690a93763949

Author: Scott Chacon <schacon@gee-mail.com>

Date: Mon Mar 17 21:52:11 2008 -0700

changed the version number

diff --git a/Rakefile b/Rakefile

index a874b73..8f94139 100644

--- a/Rakefile

+++ b/Rakefile

@@ -5,5 +5,5 @@ require 'rake/gempackagetask'

spec = Gem::Specification.new do |s|

s.name = "simplegit"

- s.version = "0.1.0"

+ s.version = "0.1.1"

s.author = "Scott Chacon"

s.email = "schacon@gee-mail.com

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 16:40:33 2008 -0700

removed unnecessary test code

diff --git a/lib/simplegit.rb b/lib/simplegit.rb

index a0a60ae..47c6340 100644

--- a/lib/simplegit.rb

+++ b/lib/simplegit.rb

@@ -18,8 +18,3 @@ class SimpleGit

end

end

-

-if $0 == \_\_FILE\_\_

- git = SimpleGit.new

- puts git.show

-end

\ No newline at end of file

该选项除了显示基本信息之外，还在附带了每次 commit 的变化。当进行代码审查，或者快速浏览某个搭档提交的 commit 的变化的时候，这个参数就非常有用了。

某些时候，单词层面的对比，比行层面的对比，更加容易观察。Git 提供了 --word-diff 选项。我们可以将其添加到 git log -p 命令的后面，从而获取单词层面上的对比。在程序代码中进行单词层面的对比常常是没什么用的。不过当你需要在书籍、论文这种很大的文本文件上进行对比的时候，这个功能就显出用武之地了。下面是一个简单的例子：

$ git log -U1 --word-diff

commit ca82a6dff817ec66f44342007202690a93763949

Author: Scott Chacon <schacon@gee-mail.com>

Date: Mon Mar 17 21:52:11 2008 -0700

changed the version number

diff --git a/Rakefile b/Rakefile

index a874b73..8f94139 100644

--- a/Rakefile

+++ b/Rakefile

@@ -7,3 +7,3 @@ spec = Gem::Specification.new do |s|

s.name = "simplegit"

s.version = [-"0.1.0"-]{+"0.1.1"+}

s.author = "Scott Chacon"

如你所见，这里并没有平常看到的添加行或者删除行的信息。这里的对比显示在行间。新增加的单词被 {+ +} 括起来，被删除的单词被 [- -] 括起来。在进行单词层面的对比的时候，你可能希望上下文（ context ）行数从默认的 3 行，减为 1 行，那么可以使用 -U1 选项。上面的例子中，我们就使用了这个选项。

另外，git log 还提供了许多摘要选项可以用，比如 --stat，仅显示简要的增改行数统计：

$ git log --stat

commit ca82a6dff817ec66f44342007202690a93763949

Author: Scott Chacon <schacon@gee-mail.com>

Date: Mon Mar 17 21:52:11 2008 -0700

changed the version number

Rakefile | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 16:40:33 2008 -0700

removed unnecessary test code

lib/simplegit.rb | 5 -----

1 file changed, 5 deletions(-)

commit a11bef06a3f659402fe7563abf99ad00de2209e6

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 10:31:28 2008 -0700

first commit

README | 6 ++++++

Rakefile | 23 +++++++++++++++++++++++

lib/simplegit.rb | 25 +++++++++++++++++++++++++

3 files changed, 54 insertions(+)

每个提交都列出了修改过的文件，以及其中添加和移除的行数，并在最后列出所有增减行数小计。 还有个常用的 --pretty 选项，可以指定使用完全不同于默认格式的方式展示提交历史。比如用 oneline 将每个提交放在一行显示，这在提交数很大时非常有用。另外还有 short，full 和 fuller 可以用，展示的信息或多或少有些不同，请自己动手实践一下看看效果如何。

$ git log --pretty=oneline

ca82a6dff817ec66f44342007202690a93763949 changed the version number

085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code

a11bef06a3f659402fe7563abf99ad00de2209e6 first commit

但最有意思的是 format，可以定制要显示的记录格式，这样的输出便于后期编程提取分析，像这样：

$ git log --pretty=format:"%h - %an, %ar : %s"

ca82a6d - Scott Chacon, 11 months ago : changed the version number

085bb3b - Scott Chacon, 11 months ago : removed unnecessary test code

a11bef0 - Scott Chacon, 11 months ago : first commit

表 2-1 列出了常用的格式占位符写法及其代表的意义。

<!-- Attention to translators: this is a table declaration. The lines must be formatted as follows <TAB><First column text><TAB><Second column text> -->

选项 说明

%H 提交对象（commit）的完整哈希字串

%h 提交对象的简短哈希字串

%T 树对象（tree）的完整哈希字串

%t 树对象的简短哈希字串

%P 父对象（parent）的完整哈希字串

%p 父对象的简短哈希字串

%an 作者（author）的名字

%ae 作者的电子邮件地址

%ad 作者修订日期（可以用 -date= 选项定制格式）

%ar 作者修订日期，按多久以前的方式显示

%cn 提交者(committer)的名字

%ce 提交者的电子邮件地址

%cd 提交日期

%cr 提交日期，按多久以前的方式显示

%s 提交说明

你一定奇怪*作者（author）*和*提交者（committer）*之间究竟有何差别，其实作者指的是实际作出修改的人，提交者指的是最后将此工作成果提交到仓库的人。所以，当你为某个项目发布补丁，然后某个核心成员将你的补丁并入项目时，你就是作者，而那个核心成员就是提交者。我们会在第五章再详细介绍两者之间的细微差别。

用 oneline 或 format 时结合 --graph 选项，可以看到开头多出一些 ASCII 字符串表示的简单图形，形象地展示了每个提交所在的分支及其分化衍合情况。在我们之前提到的 Grit 项目仓库中可以看到：

$ git log --pretty=format:"%h %s" --graph

\* 2d3acf9 ignore errors from SIGCHLD on trap

\* 5e3ee11 Merge branch 'master' of git://github.com/dustin/grit

|\

| \* 420eac9 Added a method for getting the current branch.

\* | 30e367c timeout code and tests

\* | 5a09431 add timeout protection to grit

\* | e1193f8 support for heads with slashes in them

|/

\* d6016bc require time for xmlschema

\* 11d191e Merge branch 'defunkt' into local

以上只是简单介绍了一些 git log 命令支持的选项。表 2-2 还列出了一些其他常用的选项及其释义。

<!-- Attention to translators: this is a table declaration. The lines must be formatted as follows <TAB><First column text><TAB><Second column text> -->

选项 说明

-p 按补丁格式显示每个更新之间的差异。

--word-diff 按 word diff 格式显示差异。

--stat 显示每次更新的文件修改统计信息。

--shortstat 只显示 --stat 中最后的行数修改添加移除统计。

--name-only 仅在提交信息后显示已修改的文件清单。

--name-status 显示新增、修改、删除的文件清单。

--abbrev-commit 仅显示 SHA-1 的前几个字符，而非所有的 40 个字符。

--relative-date 使用较短的相对时间显示（比如，“2 weeks ago”）。

--graph 显示 ASCII 图形表示的分支合并历史。

--pretty 使用其他格式显示历史提交信息。可用的选项包括 oneline，short，full，fuller 和 format（后跟指定格式）。

--oneline `--pretty=oneline --abbrev-commit` 的简化用法。

限制输出长度

除了定制输出格式的选项之外，git log 还有许多非常实用的限制输出长度的选项，也就是只输出部分提交信息。之前我们已经看到过 -2 了，它只显示最近的两条提交，实际上，这是 -<n> 选项的写法，其中的 n 可以是任何自然数，表示仅显示最近的若干条提交。不过实践中我们是不太用这个选项的，Git 在输出所有提交时会自动调用分页程序（less），要看更早的更新只需翻到下页即可。

另外还有按照时间作限制的选项，比如 --since 和 --until。下面的命令列出所有最近两周内的提交：

$ git log --since=2.weeks

你可以给出各种时间格式，比如说具体的某一天（“2008-01-15”），或者是多久以前（“2 years 1 day 3 minutes ago”）。

还可以给出若干搜索条件，列出符合的提交。用 --author 选项显示指定作者的提交，用 --grep 选项搜索提交说明中的关键字。（请注意，如果要得到同时满足这两个选项搜索条件的提交，就必须用 --all-match 选项。否则，满足任意一个条件的提交都会被匹配出来）

另一个真正实用的git log选项是路径(path)，如果只关心某些文件或者目录的历史提交，可以在 git log 选项的最后指定它们的路径。因为是放在最后位置上的选项，所以用两个短划线（--）隔开之前的选项和后面限定的路径名。

表 2-3 还列出了其他常用的类似选项。

<!-- Attention to translators: this is a table declaration. The lines must be formatted as follows <TAB><First column text><TAB><Second column text> -->

选项 说明

-(n) 仅显示最近的 n 条提交

--since, --after 仅显示指定时间之后的提交。

--until, --before 仅显示指定时间之前的提交。

--author 仅显示指定作者相关的提交。

--committer 仅显示指定提交者相关的提交。

来看一个实际的例子，如果要查看 Git 仓库中，2008 年 10 月期间，Junio Hamano 提交的但未合并的测试脚本（位于项目的 t/ 目录下的文件），可以用下面的查询命令：

$ git log --pretty="%h - %s" --author=gitster --since="2008-10-01" \

--before="2008-11-01" --no-merges -- t/

5610e3b - Fix testcase failure when extended attribute

acd3b9e - Enhance hold\_lock\_file\_for\_{update,append}()

f563754 - demonstrate breakage of detached checkout wi

d1a43f2 - reset --hard/read-tree --reset -u: remove un

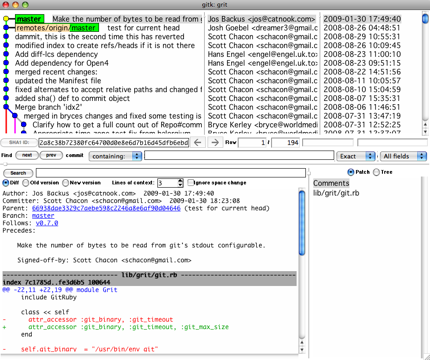
51a94af - Fix "checkout --track -b newbranch" on detac

b0ad11e - pull: allow "git pull origin $something:$cur

Git 项目有 20,000 多条提交，但我们给出搜索选项后，仅列出了其中满足条件的 6 条。

使用图形化工具查阅提交历史

有时候图形化工具更容易展示历史提交的变化，随 Git 一同发布的 gitk 就是这样一种工具。它是用 Tcl/Tk 写成的，基本上相当于 git log 命令的可视化版本，凡是 git log 可以用的选项也都能用在 gitk 上。在项目工作目录中输入 gitk 命令后，就会启动图 2-2 所示的界面。

  
 图 2-2. gitk 的图形界面

上半个窗口显示的是历次提交的分支祖先图谱，下半个窗口显示当前点选的提交对应的具体差异。