Magnum Programm Life

git引用^和~的区别

这篇git文章必转：解答我一直的疑惑

http://www.cnblogs.com/hutaoer/archive/2013/05/14/3078191.html

一. 引子

在git操作中，我们可以使用checkout命令检出某个状态下文件，也可以使用reset命令重置到某个状态，这里所说的“某个状态”其实对应的就是一个提交(commit).

我们可以把一个git仓库想象成一棵树，每个commit就是树上的一个节点。家家都有一本自己的祖谱。祖谱记录了一个家族的生命史，它不仅记录着该家族的来源、迁徙的轨迹，还包罗了该家族生息、繁衍、婚姻、文化、族规、家约等历史文化的全过程。类似的，每个git仓库都有一本自己的祖谱，仓库中commit ID的繁衍，HEAD指针的迁徙，分支的增加、更新，同样的记录着一个仓库从无到有的点点滴滴。

在git中，我们其实可以通过^和~来定位某个具体的commit，而不用每次都去敲繁琐的hash值。为了便于大家理解，先把结论放在前面：

1. “^”代表父提交,当一个提交有多个父提交时，可以通过在”^”后面跟上一个数字，表示第几个父提交，”^”相当于”^1”.
2. ~<n>相当于连续的<n>个”^”.
3. checkout只会移动HEAD指针，reset会改变HEAD的引用值。

使用git log –graph 命令，可以查看自己仓库的当前分支提交ID的树状图，如下图所示。

```
$ git log --oneline --graph
* 6652a0d Add Images for git treeview.
*   8199323 Commit A: merge B with C.
|\
| * 0cd7f2e commit C.
| |
|  \
*-. \   776c5c9 Commit B: merge D with E and F
|\ \ \
| | |/
| * |   beb30ca Commit F: merge I with J
| |\ \
| | * | 3252fcc commit J.
| * | 634836c commit I.
| |/
* | 83be369 commit E.
*   212efce Commit D: merge G with H
|\
| * 2ab52ad commit H.
* e80aa74 commit G.
```

使用git log –pretty=raw命令，可以查看commit之间的父子关系，如下图所示，需要注意的是最开始的commit是没有父提交的。

昵称：Magnum Programm Life
园龄：7年10个月
粉丝：47
关注：1
+加关注

< 2018年5月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

gdb(2)
mipi d-phy 简写 缩略词(1)
opencl(1)
video codec(1)
yuv(1)
视频格式(1)
视频解码(1)
视频压缩(1)
codec(1)
decodec(1)
更多

随笔分类

Adas(4)
Android 驱动（Linux）＆＆嵌入式(50)
Android 系统(6)
Android 應用(1)
blog(3)
camera(12)
CUDA(15)

二. 困惑

在使用git的过程中，你也许会有很多的困惑。

在使用reset或checkout命令的时候，需要一个<commit>参数，但是每次都输入commit hash值是一件比较麻烦的事情。首先你得去查询下日志，然后再用键盘将前面几位hash值输入。有时候你一次还搞不定，突然开个小差，暗恋下女神，想一想基友，都容易把hash值遗忘或弄错。肿么办？？？

又话说突然间，一堆带有hash值的符号出现在生活中，HEAD^1~4，<commit>~3^2，我擦！这是TMD玩意儿？不懂啊，使用过程中，HEAD和引用各种乱窜，根本不听从我的指挥，哎呀，妈呀！我成了git的奴隶，从此生活不再美好。肿么办？？？

不，生活还要继续，要和git做朋友。做朋友当然先要摸清楚朋友的性情和脾气咯，有了好友，生活才会充满希望。

三. 解惑

古有“射人先射马，擒贼先擒王”，今有“git仓库顺藤摸瓜”。既然commit形成的树状图，表明了各个commit之间的关系，那么我们也可以顺着这棵树去查询commit的值。一般情况下，一个commit都会有一个父提交，那么通过<commit>^这个表达式，就可以访问到其父提交的ID值；使用<commit>~也可以达到同样的功效哦。

我们知道每提交一次，HEAD就会自动移到版本库中最近的一次提交。那么HEAD^就代表了最近一次提交的父提交，HEAD~也是同样的道理；但是如果你想当然的认为^和~的用法相同，那就错了，其实它们的区别还是蛮大的。

四. 详解

我们来通过一个具体的例子，来讲解一下^和~的用法区别，同时在checkout或reset的过程中，看看HEAD和引用的变化。

查看HEAD和引用的值

我们可以通过命令来查看HEAD和引用的值，也可以通过当前仓库下的.git目录去访问。当前分支为master时，我们查看HEAD的值，命令如下：

```
$ cat .git/HEAD
ref: refs/heads/master
```

然后，我们可以查看master引用的值

```
$ cat .git/refs/heads/master
3b0370b.......  # hash code
```

master分支上初始化，并提交一次

在master分支上新建一个提交”c1”，生成commit ID 973c，这时候master引用指向973c，HEAD指向master引用。

```
$ git init
Initialized empty Git repository
$ echo c1 >> a
$ git add a
$ git commit
```

Display(6)
English learning(2)
git-sum(12)
Heterogeneous Parallel Programming(22)
life(3)
Linux && C(30)
linux 性能优化
makefile && gcc(8)
Myriad2 movidius(16)
OpenCl(13)
RenderScript
rtems and rtos(7)
Ruby
sensors(1)
shave汇编以及movidius shave(1)
Shell&&bash(11)
ubuntu&&Linux(26)
多线程(19)
感知&&opencv(2)
汇编(8)
计算机组成体系结构以及硬件介绍(12)
数据结构与算法(5)
图像，视频，解码(16)
无人机
项目管理(2)
芯片电路知识(2)
一些专业术语收集整理(4)

随笔档案

2017年12月 (7)
2017年8月 (2)
2017年2月 (1)
2017年1月 (7)
2016年12月 (2)
2016年11月 (2)
2016年10月 (1)
2016年9月 (2)
2016年8月 (1)
2016年7月 (3)
2016年6月 (4)
2016年5月 (5)
2016年4月 (1)
2016年2月 (1)
2015年12月 (9)
2015年11月 (5)
2015年10月 (6)
2015年9月 (1)
2015年8月 (6)
2015年7月 (29)
2015年6月 (12)
2015年5月 (13)
2015年4月 (17)
2015年3月 (10)
2015年2月 (7)
2015年1月 (21)
2014年12月 (25)
2014年11月 (21)
2014年10月 (33)
2014年9月 (4)
2014年8月 (10)
2014年7月 (1)
2014年6月 (6)
2014年5月 (12)
2014年4月 (1)

文章分类

```
[master (root-commit) 973c5dd] c1
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 a
$ git log --oneline
973c5dd c1
```

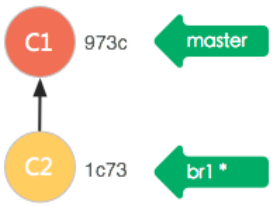
对应的图如下所示：

基于master新建br1分支，并提交两次

接下来在master分支基础上新建分支"br1"，并在"br1"上提交"c2"，commit ID为1c73，这时候HEAD指向br1，br1引用指向"c2"对应提交1c73.

```
$ git checkout -b br1
Switched to a new branch 'br1'
$ echo c2 >> b
$ git add b
$ git commit
[br1 1c7383c] c2
 1 file changed, 1 insertion(+)
 create mode 100644 b
$ git log --oneline
1c7383c c2
973c5dd c1
```

对应的图如下所示：



在分支"br1"上，提交"c3",commit ID为4927，此时HEAD指向br1，br1引用指向"c3"对应提交4927.

```
$ echo c3 >> b
$ git commit -a -m "c3"
[br1 4927c6c] c3
 1 file changed, 1 insertion(+)
$ git log --oneline
4927c6c c3
1c7383c c2
973c5dd c1
```

对应的图如下所示：

C++(2)
gdb调试(3)
视频编解码(1)

最新评论

1. Re:4.4 CUDA prefix sum一步一步优化
你好，对于第一个prefix sum的kernel有些疑惑
这种算法为什么受限制于block的大小呢？如果使用global memory的话不就可以在更大的尺度上进行计算吗？
--t800ghb
2. Re:5.1 CUDA atomic原子操作
这一篇料不太足啊，是因为原子操作尽量避免所以博主没过多去关注吗
--空空cc
3. Re:3.1 全局存储带宽与合并访问 -- Global Memory(DRAM) bandwidth and memory coalesce
内存合并访问部分中：方式B可以合并访问是因为读取了四个在global memory中不相邻的数据，但因为DRAM bursting的存在使这个数据所在的行都被读取出来，节省了其他三个thread再次读.....
--空空cc
4. Re:3.1 全局存储带宽与合并访问 -- Global Memory(DRAM) bandwidth and memory coalesce
说实话我这一部分不是太理解，我慢慢悟下，博主还有什么其他的资料吗
--空空cc
5. Re:2.2CUDA-Memory(存储)和bank-conflict
应该是"每个bank拥有1024B的数据宽度"，博主笔误了
--空空cc

阅读排行榜

1. 1.2CPU和GPU的设计区别(8527)
2. LCD显示的一些基本概念以及DSI的一些clock解释(7388)
3. 5.1 CUDA atomic原子操作(5689)
4. ubuntu64bits环境下搭建Opencl的环境(3863)
5. 集成电路中的assert和deassert应该如何翻译?(3707)

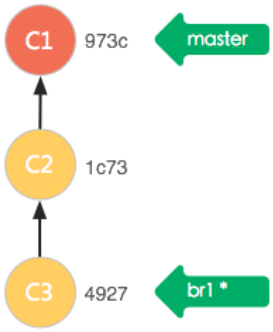
评论排行榜

1. TC358746AXBG/748XBG 桥接器说明(18)
2. MT9M021/MT9M031总结(9)
3. 1.2CPU和GPU的设计区别(5)
4. 7.OpenACC(2)
5. 3.1 全局存储带宽与合并访问 -- Global Memory(DRAM) bandwidth and memory coalesce(2)

推荐排行榜

1. 让你的Git水平更上一层楼的10个小贴士(3)
2. 1.2CPU和GPU的设计区别(3)
3. MT9M021/MT9M031总结(1)
4. MIPI DSI协议介绍(1)

5. 编译安装vim8.0(1)

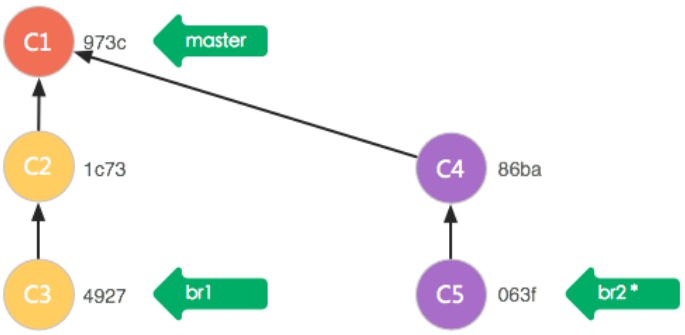


切换到master分支，基于master分支新建br2分支，并提交两次

我们先切回到master分支，然后新建分支br2，先后提交”c4”和”c5”，对应的ID分别是”86ba”和”063f”，这时候HEAD指向br2，br2引用指向”c5”的对应提交063f.git 命令如下：

```
$ git chechout master
Switched to branch 'master'
$ git checkout -b br2
Switched to a new branch 'br2'
$ echo c4 >> c
$ git add c
$ git commit -m "c4"
[br2 86ba564] c4
 1 file changed, 1 insertion(+)
 create mode 100644 c
$ git log --oneline
86ba564 c4
973c5dd c1
$ echo c5 >> c
$ git commit -a -m "c5"
[br2 063f6e6] c5
 1 file changed, 1 insertion(+)
$ git log --oneline
063f6e6 c5
86ba564 c4
973c5dd c1
```

对应的图如下所示：



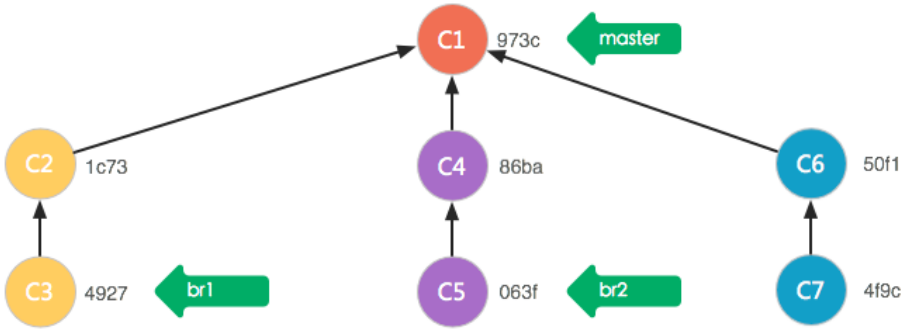
切换到master分支，基于master分支创建br3分支，并提交两次

这个操作同分支br2上类似，先从br2分支切换到master分支，然后新建分支br3，分别提交”c6”和”c7”，对应的ID分

别是"50f1"和"4f9c"，这时候HEAD指向br3，br2引用指向"c7"的对应提交4f9c，git 命令如下：

```
$ git chechout master
Switched to branch 'master'
$ git checkout -b br3
Switched to a new branch 'br3'
$ echo c6 >> d
$ git add d
$ git commit -m "c6"
[br3 50f14f6] c6
 1 file changed, 1 insertion(+)
 create mode 100644 d
$ git log --oneline
50f14f6 c6
973c5dd c1
$ echo c7 >> c
$ git commit -a -m "c7"
[br2 4f9ca79] c7
 1 file changed, 1 insertion(+)
$ git log --oneline
4f9ca79 c7
50f14f6 c6
973c5dd c1
```

对应的图如下所示：



切换到master分支，合并br1，br2和br3分支

先切换到master分支，然后合并br1 br2 br3，会新生成一个提交3b03.

```
$ git checkout master
$ git merge br1 br2 br3
 3 files changed, 6 insertions(+)
 create mode 100644 b
 create mode 100644 c
 create mode 100644 d
$ git log --oneline
3b0370b Merge braches 'br1', 'br2' and 'br3'
4f9ca79 c7
50f14f6 c6
063f6e6 c5
86ba564 c4
4927c6c c3
1c7383c c2
973c5dd c1
```

这时候，运用git log –oneline –graph查看生成的树状图，如下所示.

```
$ git log --oneline --graph
*-.   3b0370b Merge branches 'br1', 'br2' and 'br3'
|\ \
| | * 4f9ca79 c7
| | * 50f14f6 c6
| * | 063f6e6 c5
| * | 86ba564 c4
| |/
* | 4927c6c c3
* | 1c7383c c2
|/
* 973c5dd c1
```

从上图分析，在第1条红线上的commit顺序是: 3b03→4927→1c73→973c

第2条红线上的commit顺序是:3b03→063f→86ba→973c

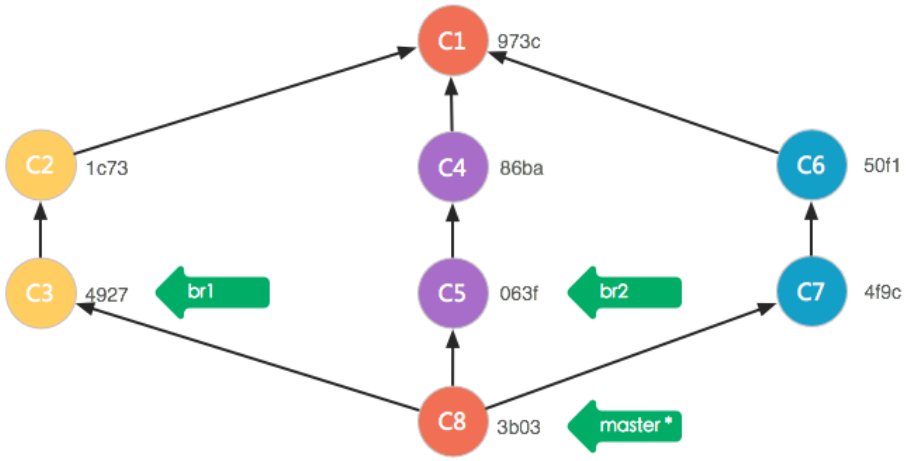
第3条黄线上的commit顺序是:3b03→4f9c→50f1→973c

这3条线的从左至右的顺序非常重要，因为HEAD^1对应的就是第1条红线的提交4927，HEAD^2对应的是第2条绿线的063f提交，HEAD^3对应的是第3条黄线的4f9c提交。3b03没有第4个父提交，因此也没有第4条线，这时候访问HEAD^n(n>3)都会报错。

因此从任何一条线上，我们都可以追溯到"c1"的commit，但是每条线上的中间节点，只能通过这条线上的节点去访问。

操作同上类似，最后的状态如下，这时候HEAD指向master，master引用指向"c8"的对应提交3b03.

对应的图如下所示：



我们再来看看3b03对应节点的父提交，如下图所示：

```
$ git log -1 --pretty=raw
commit 3b0370b942325de99eeebbfd15ff8d7d2dd66153
tree f2b770e067a0a6e0dc9f013e2465250bfcf97efa
parent 4927c6cc2b94431885d4398d0329e6894e1f195b
parent 063f6e670a068860b9ade090c3d16fa2ef9a67ad
parent 4f9ca795c81ad7127933816a94c9b2ced210aeca
author zuoyu <zuoyu.ht@taobao.com> 1367818619 +0800
committer zuoyu <zuoyu.ht@taobao.com> 1367818619 +0800

    Merge branches 'br1', 'br2' and 'br3'
```

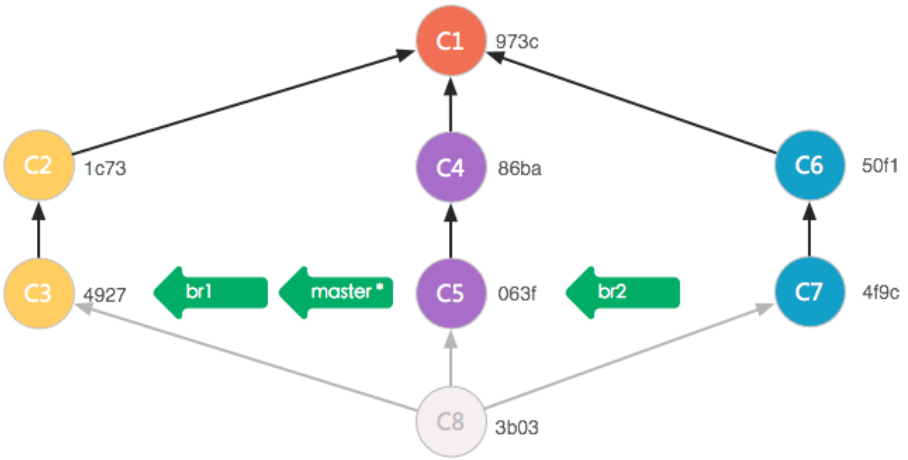

从图得知，3b03一共有三个父提交，分别是4927,063f,4f9c.

reset与checkou的区别

在master分支上，当前提交为3b03，使用git reset –hard HEAD^，将master重置到HEAD的父提交；该命令也可以写成git reset –hard HEAD^1

```
$ git reset --hard HEAD^
HEAD is now at 4927c6c c3
```

对应的图如下所示：

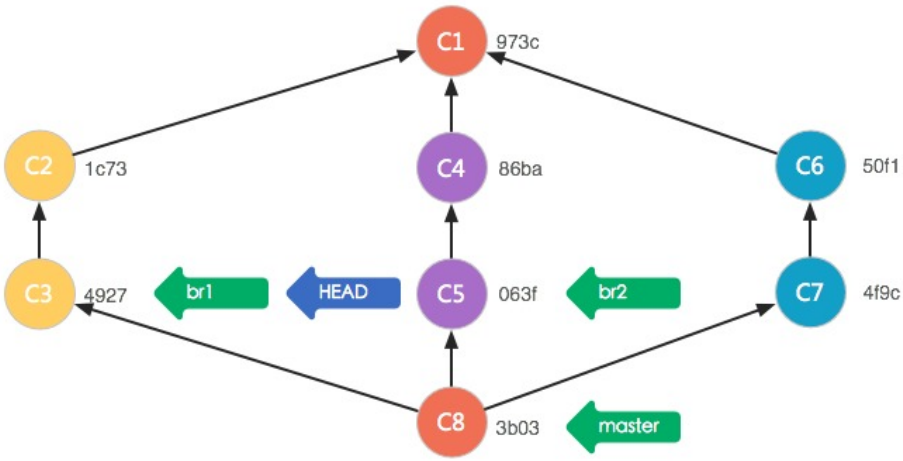


这时候，HEAD还是指向master分支，但是master引用的commit值已经变成了4927，即3b03的第一个父提交的ID.

然后，我们再重置到"c8"的commit"3b03"，git reset –hard 3b03，然后使用命令git checkout HEAD~ ，git 操作如下：

```
$ git reset --hard 3b03
HEAD is now at 3b0370b Merge branches 'br1', 'br2' and 'br3'
$ git checkout HEAD~
HEAD is now at 4927c6c... c3
```

对应的图如下所示：



这时候，HEAD指向了commit 4927，即3b03的第一个父提交ID，但是master引用还是对应的3b03.

从上面的测试，我们可以得出以下结论：

1. HEAD^,HEAD^1和HEAD~三个表达式都是代表了HEAD的父提交
2. reset <commit>的时候，HEAD不变，但是HEAD指向的引用值会变成相应的<commit>值；checkout <commit>的时候，HEAD直接变成<commit>值，但原来引用中保存的值不变。

^n和~n的区别

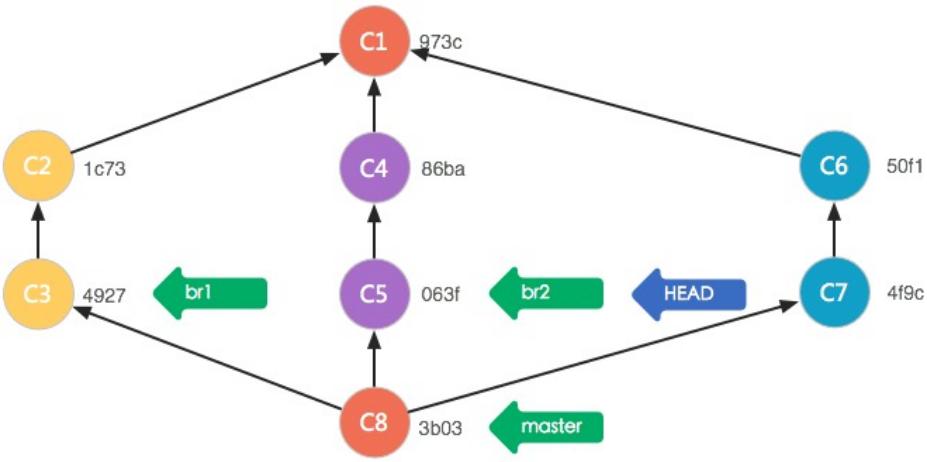
(<commit>|HEAD)^n，指的是HEAD的第n个父提交（HEAD有多个父提交的情况下），如果HEAD有N个父提交，那么n取值为n < = N.

(<commit>|HEAD)~n，指的是HEAD的第n个祖先提交，用一个等式来说明就是：(<commit>|HEAD)~n = (<commit>|HEAD)^^^….(^的个数为n).我们通过例子来验证一下吧。

我们沿用上面演示用的仓库，先检出到master分支，再使用git checkout HEAD^2，看看我们检出了哪个commit

```
$ git checkout master
$ git checkout HEAD^2
HEAD is now at 063f6e6... c5
```

我们发现"c5"对应的commit值063f正是3b03第二个父提交的commit 对应的图如下所示：

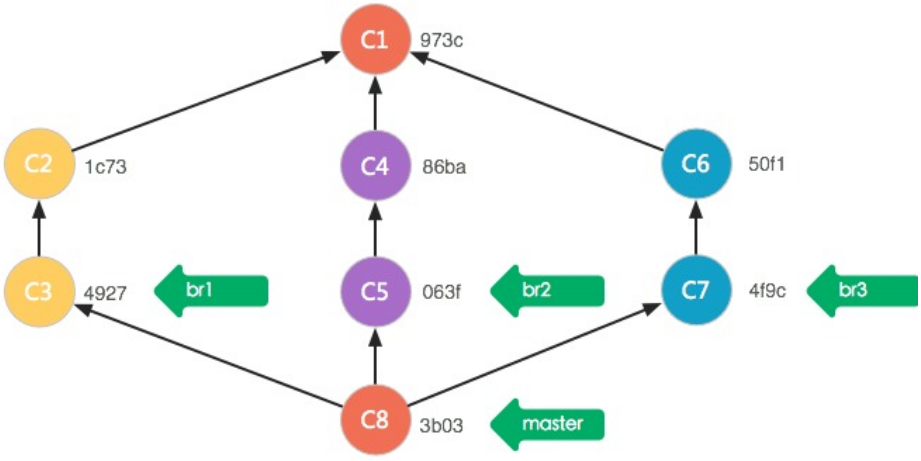


现在再切回master分支，git checkout master

然后使用git checkout HEAD^3，那么按照规律，就应该检出3b03的第三个父提交的commit，即"c7"的commit值4f9c.

```
$ git checkout master
Previous HEAD position was 063f6e6... c5
Switched to branch 'master'
$ git checkout HEAD^3
HEAD is now at 4f9ca79... c7
```

对应的图如下所示：

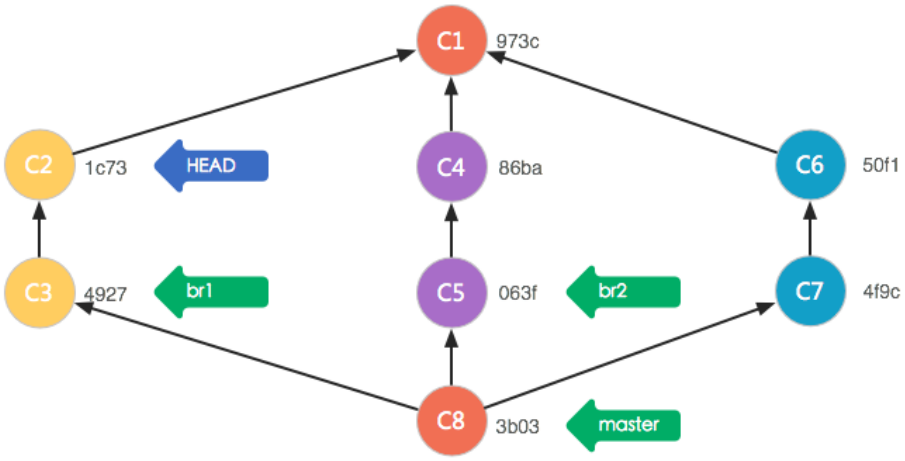


果然没错，一切都在我们的预料之中！

现在验证下HEAD~的用法，切换到master分支，然后git checkout HEAD~2

```
$ git checkout master
```

```
$ git checkout HEAD~2
HEAD is now at 1c7383c... c2
```



这时候HEAD悄然来到了"c2"的commit 1c73，因此，HEAD~2 相当于HEAD的第一个父提交的第一个父提交。即HEAD~2 = HEAD^^ = HEAD^1^1, 符合预期！好开心的哟！

五.总结

1. “^”代表父提交,当一个提交有多个父提交时，可以通过在"^"后面跟上一个数字，表示第几个父提交，"^"相当于"^1".
2. ~<n>相当于连续的<n>个"^".
3. checkout只会移动HEAD指针，reset会改变HEAD的引用值。

现在看到^和~两个符号，再也不会彷徨和害怕了，因为我们知道了它们之间的关系及区别，从此我们过上了幸福的生活。

分类: git-sum

Magnum Programm Life
关注 - 1
粉丝 - 47
+加关注

« 上一篇：Git log高级用法
» 下一篇：Git提交引用和引用日志

posted @ 2015-12-28 16:58 Magnum Programm Life 阅读(227) 评论(0) 编辑 收藏

最新IT新闻:
· 英特尔10nm制程量产之痛，目标有些“激进”
· 用外卖业务反击美团，可能是滴滴的一个战略错误
· 中国电影票房造假史：这是场制片、发行、院线的权力游戏
· 数据丑闻不会影响Facebook和扎克伯格前进的脚步
· Instagram重磅更新 全新设计探索页面增加视频聊天功能
» 更多新闻...

最新知识库文章:
· 如何识别人的技术能力和水平？
· 写给自学者的入门指南
· 和程序员谈恋爱
· 学会学习
· 优秀技术人的管理陷阱
» 更多知识库文章...