

mliubing2532的专栏

目录视图 摘要视图 RSS 订阅

个人资料



生命不息奋斗不止

访问：22913次  
积分：409  
等级：BLOG 2  
排名：千里之外  
原创：18篇 转载：3篇  
译文：0篇 评论：5条

文章搜索

文章分类

- 心情随笔 (1)
- Ubuntu使用总结 (3)
- Linux相关 (6)
- Android相关 (3)
- linux工具命令的使用 (4)
- git使用总结 (3)
- adb相关 (2)
- Linux drivers (1)
- C/C++经验总结 (1)
- STM32系列 (1)
- ARM汇编 (0)

文章存档

- 2013年03月 (1)
- 2012年08月 (1)
- 2012年06月 (1)
- 2012年05月 (18)

阅读排行

- git-rebase用法总结 (4575)
- git-am用法 (4545)
- adb-Android debug bridg (2575)
- Android source build/env (1642)
- git-rebase(认真看，分析 (1304)
- g\_sensor工作原理 (1094)
- Ubuntu/环境变量:修改/et (1038)
- 原子上下文 (882)
- 小知识点(备查) (411)
- C中extern关键字详解 (303)

Markdown博文大赛清新开启 天天爱答题 一大波C币袭来 寻找Java大牛！ 一次拿下软考，我自有锦囊妙计！

git-rebase(认真看，分析很到位)

分类：linux工具命令的使用 git使用总结 2012-05-17 21:21 1304人阅读 评论(3) 收藏 举报

branch git c patch merge tree

命令格式

我们先来看看git-rebase的命令格式：

git rebase [-i | --interactive] [options] [--onto <newbase>] <upstream> [<branch>]  
git rebase [-i | --interactive] [options] --onto <newbase> --root [<branch>]  
git rebase --continue | --skip | --abort

从命令格式，可以看到git-rebae命令至少需要一个参数，那就是<upstream>，这个参数可以是一个分支名称，也可以是一次有效的commit。

一个小地方

在你决定学习这个命令，首先有一个小地方你应该注意，那就是如果git-rebase后面加上了参数<branch>，那么 git-rebase会在任何其他动作之前先执行git checkout <branch>，如果没有加参数<branch>，那么git-rebase会针对当前分支来做动作。

命令用处

git-rebase命令主要用在从上游分支获取最新commit信息，并有机的将当前分支和上游分支进行合并。

这是只言片语的介绍，可能读者并不能了解它的用途和好处。

还是要用例子说话。我们假设主分支为master，在开发过程中生成一个新分支topic。master称为topic的上游分支。

例子开始：

```
[roccrocket@abc git-study]$ cd rebase
[roccrocket@abc rebase]$ ls
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ cat roc.c
int main()
{
    printf("master:001");
    return 0;
}
[roccrocket@abc rebase]$ git init
Initialized empty Git repository in /roccrocket/career/programming/git-study/rebase/.git/
[roccrocket@abc rebase]$ git add .
[roccrocket@abc rebase]$ git commit -m "master:001"
Created initial commit 2d89602: master:001
1 files changed, 5 insertions(+), 0 deletions(-)
create mode 100644 roc.c
[roccrocket@abc rebase]$ git log
commit 2d89602d0c9955824df0d2c023e447f5d98d863a
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:26:40 2008 +0800
```

评论排行

git-rebase(认真看，分析	(3)
Ubuntu/环境变量:修改/etc	(1)
git-rebase用法总结	(1)
开博第一篇	(0)
android 代码结构	(0)
g_sensor工作原理	(0)
Emacs常用命令	(0)
自旋锁和信号量	(0)
platform_device&&platfo	(0)
C中extern关键字详解	(0)

推荐文章

- \* 【ShaderToy】开篇
- \* FFmpeg源代码简单分析：avio\_open2()
- \* 技能树之旅: 从模块分离到测试
- \* Qt5官方demo解析集36——Wiggly Example
- \* Unity3d HDR和Bloom效果（高动态范围图像和泛光）
- \* Android的Google官方设计指南

最新评论

- git-rebase用法总结  
yamakasiluke: 谢谢楼主,懂了,只push过,rebase没有处理过conflict.
- git-rebase(认真看，分析很到位)  
vc2009: @SLZ644644:谢谢，现在已然懂了
- git-rebase(认真看，分析很到位)  
左耳兔: git rebase -i HEAD~2
- Ubuntu/环境变量:修改/etc/envirc  
紫梦lan: 学习了
- git-rebase(认真看，分析很到位)  
vc2009: git-rebase 如何合并两次提交呢？

master:001

[roccrocket@abc rebase]\$

到此，我们已经在master分支完成了一个commit。

```
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ git commit -a -m "master:002"
Created commit 41b3d1c: master:002
1 files changed, 1 insertions(+), 0 deletions(-)
[roccrocket@abc rebase]$ cat roc.c
int main()
{
    printf("master:001");
    printf("master:002");
    return 0;
}
```

```
[roccrocket@abc rebase]$ git log
commit 41b3d1cfaae0184bb8e5f27a165d51cc23867413
Author: roccrocket <wupengchong@gmail.com>
Date: Mon Nov 17 15:28:01 2008 +0800
```

master:002

```
commit 2d89602d0c9955824df0d2c023e447f5d98d863a
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:26:40 2008 +0800
```

master:001

[roccrocket@abc rebase]\$

到此为止，我们已经在master分支完成了两次commit的提交。

现在的分支结构是这样的，请读者记清楚：

master:001 — master:002 (master)

好，我们继续要做的事情就是建立一个新的分支topic。

```
[roccrocket@abc rebase]$ git branch
* master
[roccrocket@abc rebase]$ git branch topic
[roccrocket@abc rebase]$ git branch
* master
topic
[roccrocket@abc rebase]$ git checkout topic
Switched to branch "topic"
[roccrocket@abc rebase]$ git branch
master
* topic
[roccrocket@abc rebase]$
```

好了，我们已经成功建立了topic分支，并且已经转移到了topic分支。

接下来，topic上面的开发情况如下：

```
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ git commit -a -m "topic:001"
Created commit d599b54: topic:001
1 files changed, 1 insertions(+), 0 deletions(-)
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ git commit -a -m "topic:002"
Created commit 3f4b17f: topic:002
1 files changed, 1 insertions(+), 0 deletions(-)
[roccrocket@abc rebase]$ cat roc.c
int main()
{
```

```

printf("topic :002");
printf("topic :001");
printf("master:001");
printf("master:002");
return 0;
}
[roccrocket@abc rebase]$ git log
commit 3f4b17fe3b5d277771770c0515e75f04e783ad14
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:49:24 2008 +0800

topic:002

commit d599b54336ad96b8e09ef92e371a09a25e6d0c11
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:48:58 2008 +0800

topic:001

commit 41b3d1cfcae0184bb8e5f27a165d51cc23867413
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:28:01 2008 +0800

master:002

commit 2d89602d0c9955824df0d2c023e447f5d98d863a
Author: roccrocket <abc@gmail.com>
Date: Mon Nov 17 15:26:40 2008 +0800

master:001

```

```
[roccrocket@abc rebase]$
```

可知，自从建立并切换到topic分支后，topic又进行了两次commit提交，每次分别加入了一行代码。

此时，分支结构应该是这样的：

topic:001 --- topic:002 ( topic ) / master:001 --- master:002 (master)

这个图已经很清晰了，可以看出分支的走向。

接下来，master分支也有自己的进度，如下：

```

[roccrocket@abc rebase]$ git checkout master
Switched to branch "master"
[roccrocket@abc rebase]$ git branch
* master
topic
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ git commit -a -m "master:003"
Created commit 91a7ffc: master:003
1 files changed, 1 insertions(+), 0 deletions(-)
[roccrocket@abc rebase]$ vi roc.c
[roccrocket@abc rebase]$ git commit -a -m "master:004"
Created commit b81fbc3: master:004
1 files changed, 1 insertions(+), 0 deletions(-)
[roccrocket@abc rebase]$ cat roc.c
int main()
{
printf("master:001");
printf("master:002");
printf("master:003");
printf("master:004");
return 0;
}
[roccrocket@abc rebase]$ git log

```

```
commit b81fbc3be5c7bd1fdef72820c29e2c67590f4b03
```

```
Author: rocrocket <wupengchong@gmail.com>
```

```
Date: Mon Nov 17 15:55:23 2008 +0800
```

```
master:004
```

```
commit 91a7ffc73e6320a86b10849061efd672f47fd5bd
```

```
Author: rocrocket <abc@gmail.com>
```

```
Date: Mon Nov 17 15:55:06 2008 +0800
```

```
master:003
```

```
commit 41b3d1cfaae0184bb8e5f27a165d51cc23867413
```

```
Author: rocrocket <abc@gmail.com>
```

```
Date: Mon Nov 17 15:28:01 2008 +0800
```

```
master:002
```

```
commit 2d89602d0c9955824df0d2c023e447f5d98d863a
```

```
Author: rocrocket <abc@gmail.com>
```

```
Date: Mon Nov 17 15:26:40 2008 +0800
```

```
master:001
```

```
[rocrocket@abc rebase]$
```

可以看到，master分支也完成了两次commit提交，每次分别添加了一行代码。

截止此时，分支结构为：

```
topic:001 --- topic:002 ( topic ) / master:001 --- master:002 --- master:003 ---
master:004 (master)
```

在这个时候，我们的实验样本已经基本搭建完毕，git-rebase就要派上用场了！

我们假设topic和master是一个项目的两个分支，master当然是主分支，而topic是旁路分支。在软件开发的大部分情况中，旁路分支是要遵从主分支的。所以说，现在topic分支想将master分支开发的最新代码导入到topic分支中，而且要求此动作不影响master主分支的开发，也就是说要暗中完成。git-rebase上场了：

```
[rocrocket@abc rebase]$ git checkout topic
```

```
Switched to branch "topic"
```

```
[rocrocket@abc rebase]$ git branch
```

```
master
```

```
* topic
```

```
[rocrocket@abc rebase]$ git rebase master
```

```
First, rewinding head to replay your work on top of it...
```

```
Applying topic:001
```

```
error: patch failed: roc.c:1
```

```
error: roc.c: patch does not apply
```

```
Using index info to reconstruct a base tree...
```

```
Falling back to patching base and 3-way merge...
```

```
Auto-merged roc.c
```

```
Applying topic:002
```

```
[rocrocket@abc rebase]$
```

我们使用了git rebase master来完成我们的需求。如果你心细的话，你会看到它输出了一些error，意思是说“补丁失败”，这个error没有关系，不影响git-rebase的正常工作的。

让我们来看看git-rebase的魔力吧：

```
[rocrocket@abc rebase]$ git branch
```

```
master
```

```
* topic
```

```
[rocrocket@abc rebase]$ git log
```

```
commit 05de9849078541c86cf5182cd8c15fa22bd00f77
```

```
Author: rocrocket <abc@gmail.com>
```

```
Date: Mon Nov 17 15:49:24 2008 +0800
```

```
topic:002
```

commit 7e5a744ef9e0740b4a091e9c8baa859b14800b0b

Author: rocrocket <abc@gmail.com>

Date: Mon Nov 17 15:48:58 2008 +0800

topic:001

commit b81fbc3be5c7bd1fdef72820c29e2c67590f4b03

Author: rocrocket <abc@gmail.com>

Date: Mon Nov 17 15:55:23 2008 +0800

master:004

commit 91a7ffc73e6320a86b10849061efd672f47fd5bd

Author: rocrocket <abc@gmail.com>

Date: Mon Nov 17 15:55:06 2008 +0800

master:003

commit 41b3d1cfaae0184bb8e5f27a165d51cc23867413

Author: rocrocket <abc@gmail.com>

Date: Mon Nov 17 15:28:01 2008 +0800

master:002

commit 2d89602d0c9955824df0d2c023e447f5d98d863a

Author: rocrocket <abc@gmail.com>

Date: Mon Nov 17 15:26:40 2008 +0800

master:001

[rocrocket@abc rebase]\$ cat roc.c

```
int main()
{
    printf("topic :002");
    printf("topic :001");
    printf("master:001");
    printf("master:002");
    printf("master:003");
    printf("master:004");
    return 0;
}
```

[rocrocket@abc rebase]\$

看到了吧！master分支刚才开发的master:003和master:004也已经悄悄的进入了topic分支的日志里了。而在roc.c文件中也有了相应的开发代码。

这下，你是不是豁然开朗的感觉？呵呵，来看一下分支结构图：

master:001 --- master:002 --- master:003 --- master:004 --- topic:001 --- topic:002  
(topic)

master:001 --- master:002 --- master:003 --- master:004(master)

这就是git-rebase的魔力！看出神奇之处了么？

如果忘了，就对比一下。这是执行git-rebase之前的分支结构图：

topic:001 --- topic:002 (topic) / master:001 --- master:002 --- master:003 --- master:004 (master)

上一篇 git-rebase用法总结

下一篇 git-am用法

主题推荐 git 软件开发 结构 工作 合并

猜你在找

- The git rebase Command

git rebase用法解析

git 关于merge rebase衍合

git rebase 使用
- 【精品课程】HTML+CSS网页设计由浅入深

【精品课程】Extjs在Asp.Net中的应用开发

【精品课程】Java Swing、JDBC开发桌面级应用

【精品课程】ASP.NET下工作流技术WorkFlow4.0应用

准备好了么？跳 吧！


更多职位尽在 CSDN JOB

数据分析开发工程师	我要跳槽	数据分析研究员	我要跳槽
杭州行云信息科技有限公司	3-15K/月	上海维赛特科技实业有限公司	4-8K/月
Hadoop数据分析师	我要跳槽	系统分析师	我要跳槽
深圳市聚领信息科技有限公司	10-15K/月	西安赛思通软件技术有限公司	3-15K/月




查看评论

2楼 左耳兔 2012-11-14 13:16发表




git rebase -i HEAD~2

Re: vc2009 2013-06-09 09:43发表



回复SLZ644644：谢谢，现在已然懂了

1楼 vc2009 2012-08-17 09:00发表



git-rebase 如何合并两次提交呢？

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
- VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
- BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
- Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
- coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
- Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
- Angular Cloud Foundry Redis Scala Django Bootstrap