**Repo命令使用说明**

公司：深圳市宝能投资集团有限公司

版本说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 版本号 | 日期 | 作者 |
| 1 | V1.0 | 2020.10.21 | 刘蓉 |
| 2 |  |  |  |
| 3 |  |  |  |

[1.repo init 3](#_Toc436160364)

[2.repo sync 4](#_Toc436160365)

[3.repo start 5](#_Toc436160366)

[4.repo checkout 5](#_Toc436160367)

[5.repo status 5](#_Toc436160368)

[6.repo upload 6](#_Toc436160369)

[7.repo branches 6](#_Toc436160370)

[8.repo diff 7](#_Toc436160371)

[9.repo stage –i 7](#_Toc436160372)

[10.repo rebase 9](#_Toc436160373)

[11.repo prune 10](#_Toc436160374)

[12.repo abandon 10](#_Toc436160375)

[13.repo forall 10](#_Toc436160376)

声明：下面的repo都是本地配置的repo引导脚本的名称

# 1.repo init

repo init子命令主要完成检出清单版本库(manifests.git)，以及配置Git用户名和邮件地址的工作

举例：

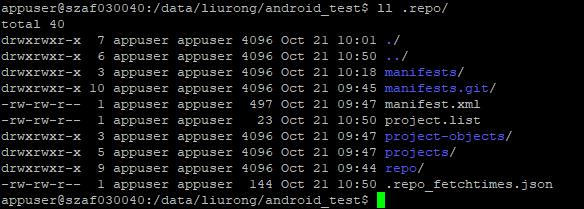
$cd working-tree

$repo init –u ssh://xxx@ip:29418/manifests.git –b branchname –m project.xml

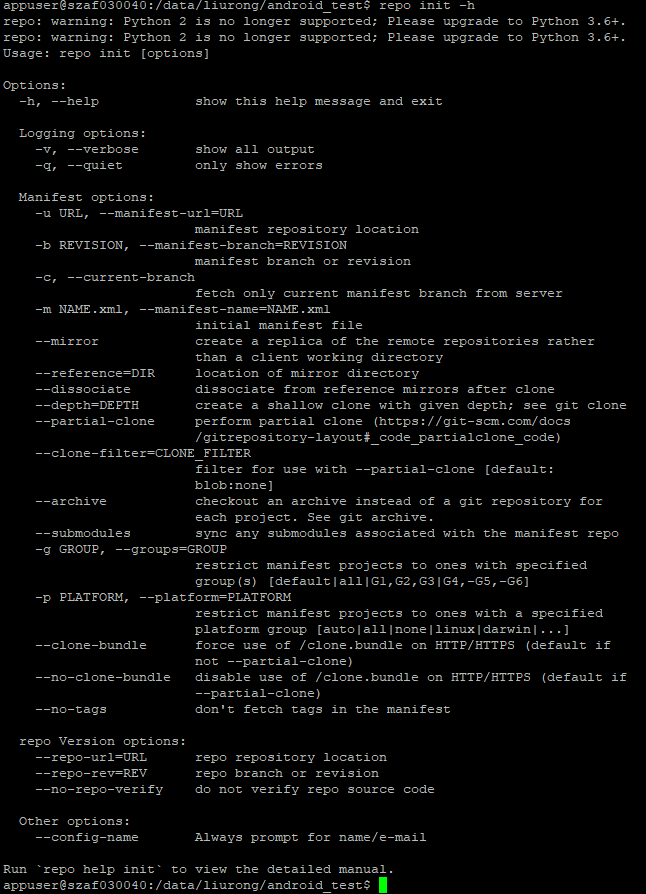
指定以清单库中的project.xml文件完成repo在本地的初始化

$ls –lah

$cd .repo



repo init的参数



# 2.repo sync

下载代码或者同步代码

相当于git clone 或者git remote update git rebase origin/branchname

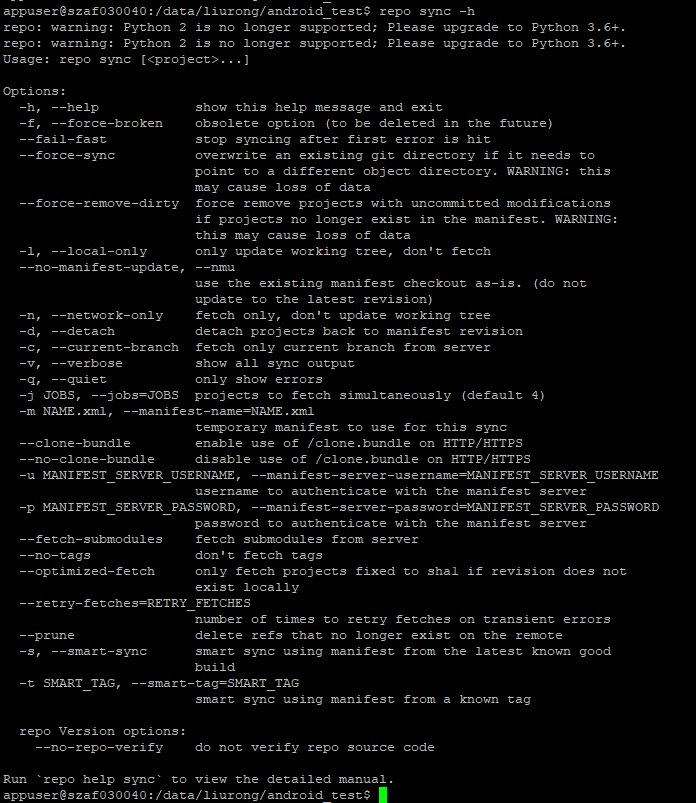
注意: **采用repo sync下载到本地的代码库，没有默认的本地分支，即不会检出任何分支，需要通过 repo start 命令创建本地库分支，用于追踪远程库对应的分支**

举例: $ cd working-tree

$repo sync

如果是首次执行此命令，可以看到对应项目的所有模块的代码库都被拉到working-tree工作目录下

repo sync 参数



# 3.repo start

是对git checkout -b 的封装

可以利用repo start新建本地分支用于追踪远程分支

举例: $ cd working-tree

$ repo start develop --all (在某个项目的所有的模块代码库中建立develop分支)

如果只想在某几个模块库中建立分支：

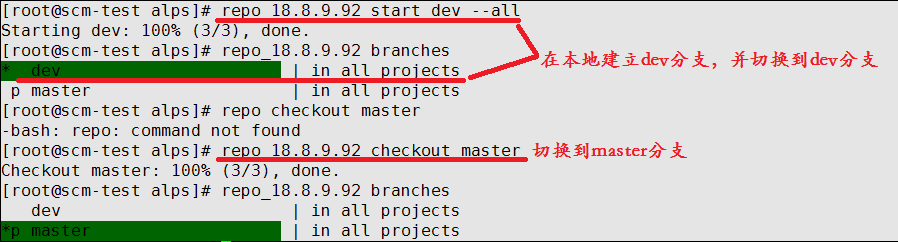
$repo start develop 模块库的名称 模块库的名称 模块库的名称

# 4.repo checkout

用法:repo checkout branchname <project>

是对git chekcout的封装，检出之前，需要利用repo start创建分支

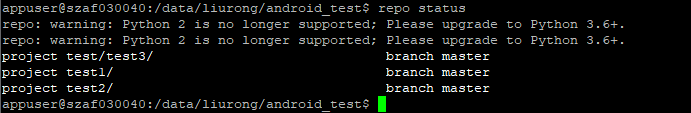
举例:



# 5.repo status

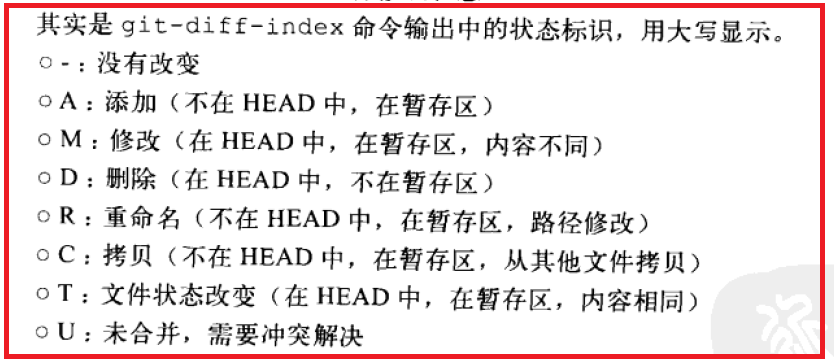
是对git diff-index、git diff-files命令的封装，显示暂存区的状态和本地文件修改的状态

举例:

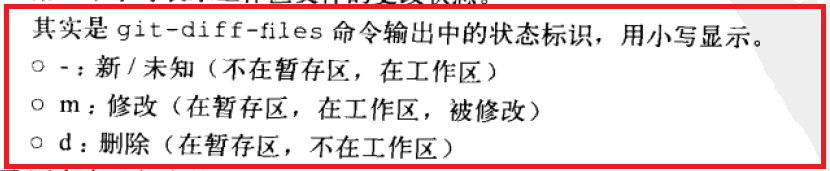


执行$repo status 之后的打印信息中，头两个字母表示变更的状态，后面显示文件名或其他变更信息，如上图所示

**处于第一个字符的字母：表示暂存区的文件修改状态**



**处于第二个字符的字母：表示工作区的文件修改状态**

****

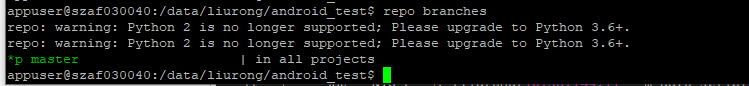
# 6.repo upload

repo upload 相当于git push ,但是又有很大的不同。执行repo upload不是将版本库改动推送到克隆是的远程服务器，而是推送到代码审查服务器（由Gerrit软件架设)的特殊引用上，使用的ssh协议(特殊端口),代码审查服务器会对推送的提交进行特殊处理，将新的提交显示为一个待审核的修改集，并进入代码审核流程。只有当审核通过后，才会合并到官方正式的版本库中

# 7.repo branches

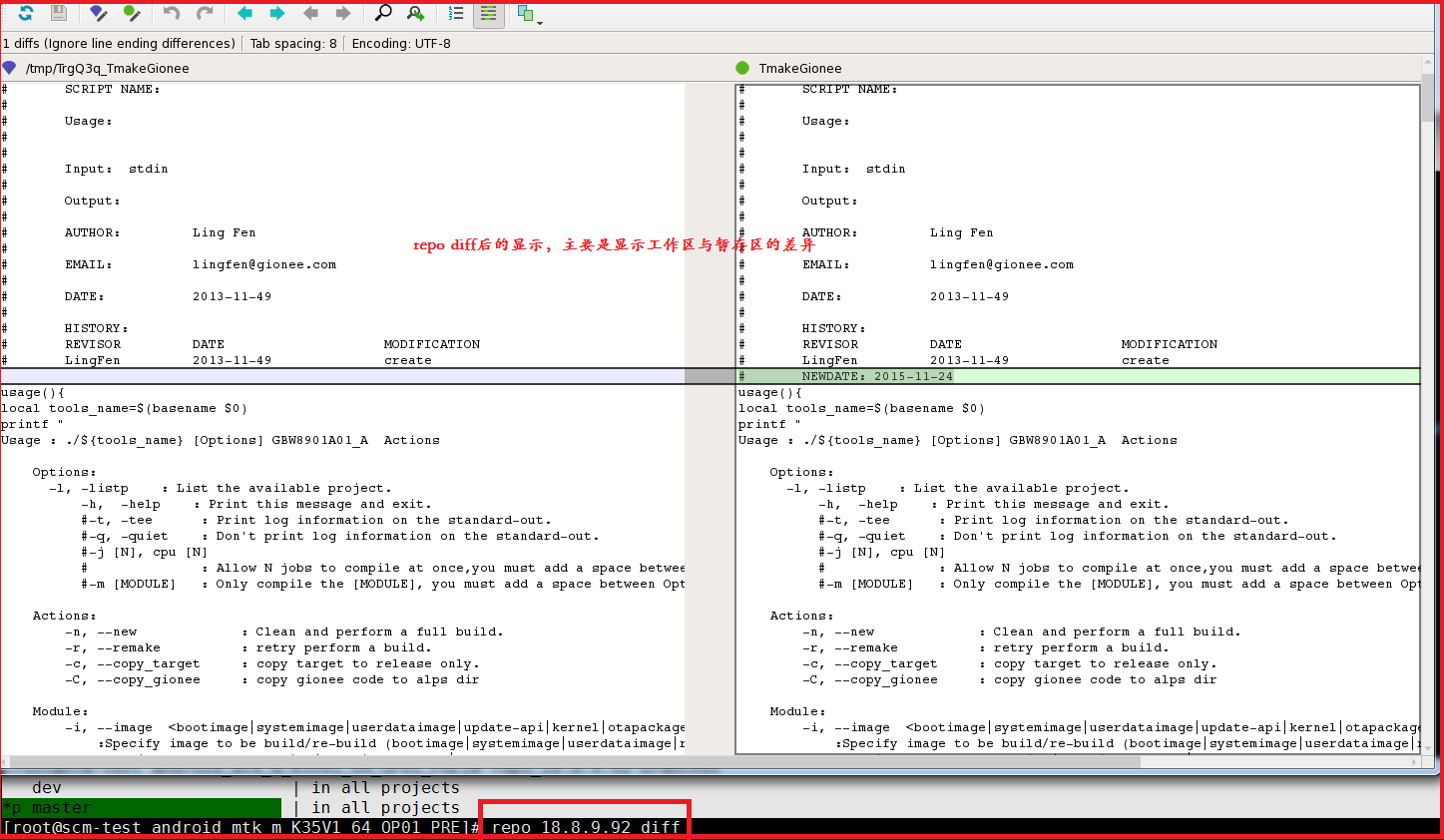
读取各个子库的本地分支列表并汇总显示。该命令实际上通过直接读取.git/refs目录下的引用来获取分支列表，以及分支的发布状态。

举例：



# 8.repo diff

是对git diff的封装，用于分别显示各个子项目库工作区与暂存区的差异



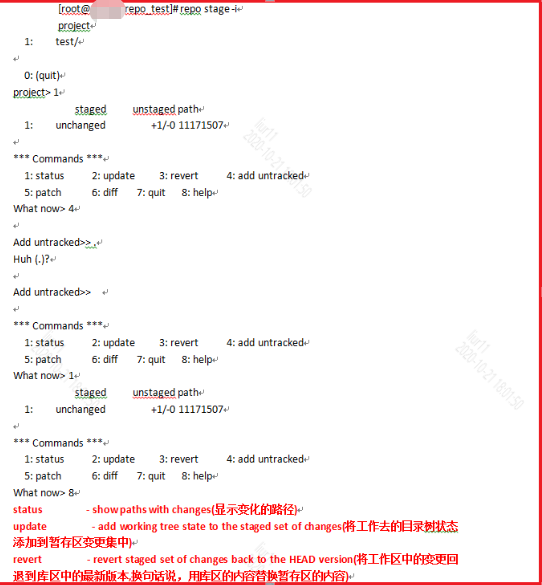
# 9.repo stage –i

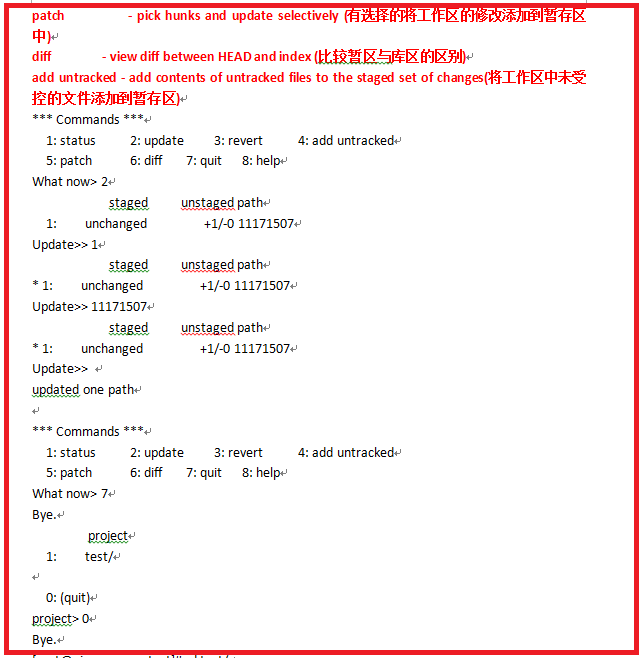
repo stage 子命令实际上是对git add --interactive命令的封装，用于挑选各个子项目库工作区中的改动以加入暂存区

用法：

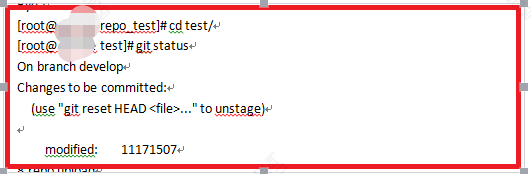
repo statge -i <project>

举例：





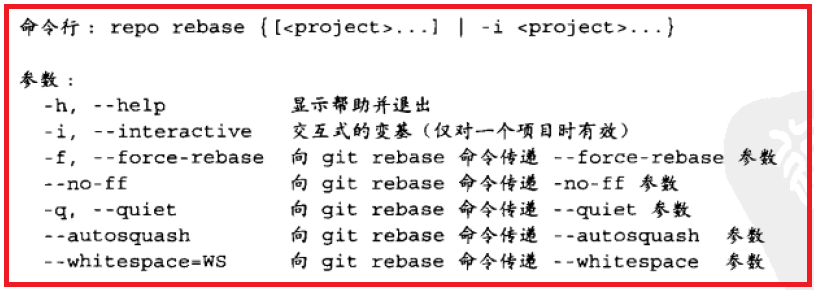
使用git命令验证



# 10.repo rebase

repo rebase是对git rebase命令的封装，该命令的参数也作为git rebase命令的参数，但-i参数仅对当对一个项目子库执行时才有效

用法:



# 11.repo prune

Repo prune 是对git branch –d 的封装，该命令用于扫描项目的各个分支，并删除已经合并的分支

用法：Repo prune <project>

# 12.repo abandon

Repo abandon 是对git branch –D的封装，注意在使用这条命令之前，请确保被删除的分支上的代码都已经合并

用法:Repo abandon <branchname> <project>

# 13.repo forall

**迭代器，可以对repo管理的所有子项目库进行迭代**

**最常用的命令:$repo forall –c “git所有的命令” <project>**

**$repo forall -c “其他的系统命令” <project>**

**如果不指定projectname，会对repo管理的子库进行迭代操作**