1.repo

repo是一个脚本。用来管理由多个git仓库组成项目的脚本工具。

通过一个标有远端地址，分支信息，仓库名等信息的xml文件去管理多个git仓库组成的项目。

2.repo下载

摘自google android官网

To install, initialize, and configure Repo, follow these steps:

Make sure you have a bin/ directory in your home directory, and that it is included in your path:

$ mkdir ~/bin  
$ PATH=~/bin:$PATH

Download the Repo script and ensure it is executable:

$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo  
$ chmod a+x ~/bin/repo

3.核心是manifest文件，让我们看一下

<?xml version="1.0" encoding="UTF-8"?>

<manifest>

<remote name="aosp"

fetch=".." /> //这里声明了一个叫aosp的remote，地址是'..’，意思是这个menifest的地址的上一级，我们看下这个menifest的地址

<https://android.googlesource.com/platform/manifest>，所以fetch地址就是https://android.googlesource.com

//之前老版本的repo，fetch是不支持相对路径的。

<default revision="refs/tags/android-4.0.3\_r1"

remote="aosp"

sync-j="4" /> //声明了一些默认参数，分支，remote等，sync-j参数是后来新加的支持，应该是repo sync的默认线程数，可以加快sync的速度

<project path="build" name="platform/build">

<copyfile src="core/root.mk" dest="Makefile" />

</project>//声明了一个project,path="build" name="platform/build"表示会从https://android.googlesource.com/platform/build.git仓库clone到本地，checkout后会建立一个build目录，copyfile参数表示会在checkout的过程结束后，把src制定的文件拷贝到dest制定的地方，这里是把build/core/root.mk文件拷贝到源码根目录，取名Makefile。可以思考下为什么这样做。

<project path="abi/cpp" name="platform/abi/cpp" />//剩下的就与上面类似了，只不过目录结构会多一些，这里就是把https://android.googlesource.com/platform/abi/cpp.git仓库clone到本地，并checkout到“本地android源码根目录/abi/cpp”中。

<project path="bionic" name="platform/bionic" />

<project path="bootable/bootloader/legacy" name="platform/bootable/bootloader/legacy" />

<project path="bootable/diskinstaller" name="platform/bootable/diskinstaller" />

<project path="bootable/recovery" name="platform/bootable/recovery" />

<project path="cts" name="platform/cts" />

<project path="dalvik" name="platform/dalvik" />

<project path="development" name="platform/development" />

<project path="device/common" name="device/common" />

?xml version="1.0" encoding="UTF-8"?>

…...............

后面省略若干行。。。

3.小例子

自己搭建一个由三个仓库组成的项目，用repo管理起来。

$mkdir source.git

$cd source.git

$mkdir manifest.git //建立我们manifest文件存放的仓库目录

$mkdir project1.git

$mkdir dir1

$cd dir1

$mkdir project2.git

$mkdir project3.git

//建立第一个仓库

$cd ../project1.git

$touch project1.cpp

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit” //git add . 与commit -am不是什么太好的习惯，add .很容易把不该跟踪的东西加入到版本库中，commit -am 则让你失去最后一次检查到这个错误的机会，好习惯是挨个git add你想要提交的文件，然后commit,尽量不要使用-m那样说明你的提交注释非常少。

//建立第二个与第三个仓库

$cd ../dir1/project2.git

$touch project2.cpp

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

$cd ../project3.git

$touch project3.cpp

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

//建立manifest仓库

$cd ../../manifest.git

$gedit default.xml //额，我故意没有用vi或者emacs，这算不算增进用户体验？

//添加如下内容

<?xml version="1.0" encoding="UTF-8"?>

<manifest>

<remote name="test-remote"

fetch="." />

<default revision="branch-test"

remote="test-remote"

sync-j="4" />

<project path="project1" name="project1" />

<project path="dir1/project2" name="dir1/project2" />

<project path="dir1/project3" name="dir1/project3" />

</manifest>

//提交

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

这里我们就把三个仓库与manifest仓库都建好了，然后我们来使用repo

新建一个目录来保存我们的项目

$mkdir repo\_test

$cd repo\_test

$repo init -u ~/source.git/manifest.git -b branch-test

//确认配置

//执行完毕后开始同步代码

$repo sync

这样大家就可以从这个小例子看出如何使用repo去管理你自己的代码了。

4.repo的使用

