Repo

repo为了方便管理git库而开发的python脚本。

.repo/manifests

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

<manifest>

<remote name="aosp"

fetch=".."

review="https://android-review.googlesource.com/" />

<default revision="master"

remote="aosp"

sync-j="4" />

<project path="build" name="platform/build" groups="pdk,tradefed" >

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

</project>

<project path="abi/cpp" name="platform/abi/cpp" groups="pdk" />

<project path="art" name="platform/art" groups="pdk" />

...

<project path="tools/studio/translation" name="platform/tools/studio/translation" groups="notdefault,tools" />

<project path="tools/swt" name="platform/tools/swt" groups="notdefault,tools" />

</manifest>

<remote>：描述远程仓库信息，name描述名称（xml属性），fetch项目名前缘，review：审查的地址

<default>：定义属性，project标签的默认属性，属性revision表示当前版本，sync -j 表示同步远程代码时，并发任务数量。

<project>：每个repo管理的git库，就对应到每个project标签，path描述项目相对远程仓库url的路径。同时将作为对应git库在本地代码路径，name定义项目名称。

修改：

假如需要修改替换一些git库，需要修改default.xml。但直接定制会导致下一个更新项目清单造成冲突。repo提供了一个灵活方式local\_manifest:所有定制是遵循default.xml规范。文件可自定义。将新建的.repo/local\_manifests子目录即可。repo会遍历 .repo/local\_manifests目录下的所有\*.xml文件，最终与default.xml合并成一个总项目清单文件manifest.xml。

REPO脚本库：.repo/repo

repo是对git封装，提供一套repo命令集，init和sync等，repo自动化实现也都在git库里面，在第一次初始化时，repo会远程下载git库到本地。

仓库目录和工作目录：

仓库目录保存历史信息，修改记录等，工作目录保存的是当前版本信息。git仓库目录位于工作目录下，但git支持仓库和工作目录分开放。repo管理既有分开，也有在工作目录下。