1. 任务描述

我在58集团安居客实习，主要从事iOS开发，由于实习时间长，完成任务较多，包括但不限于以下几个方面：

- 学习xcode使用，熟悉iOS开发
- 学习cocoapods集成三方库的方法
- 学习git版本控制系统的使用，熟悉团队合作的机制
- 熟悉公司代码，进行实际的小规模个人开发
- 参与公司项目开发，实现产品需求

2. 技术性工作

2.1 cocoapods集成第三方库

- 因为cocoapods由ruby编写，因此少不了ruby环境，首先打开终端查看当前ruby版本，版本大于等于2.2.2即可，否则点击这里查看详细信息

```
ruby -v

//我的版本如下
ruby 2.3.7p456 (2018-03-28 revision 63024) [universal.x86_64-darwin17]
```

- 因为ruby源国内被墙，因此将ruby源更换为ruby-china（淘宝源也已停止维护）

```
gem sources --remove https://rubygems.org/
gem sources --add https://gems.ruby-china.org/
```

- 验证ruby源是且仅是ruby-china,执行如下命令查看

```
gem sources -l
//得到如下结果即为正确

*** CURRENT SOURCES ***

https://gems.ruby-china.org/
```

- 开始安装cocoapods

```
sudo gem install -n /usr/local/bin cocoapods
```

- 安装本地库

```
pod setup
```

- 执行结果,等待过程较为漫长

```
Setting up CocoaPods master repo
$ /usr/bin/git clone https://github.com/CocoaPods/Specs.git master --progress
Cloning into 'master'...
remote: Counting objects: 1879515, done.
remote: Compressing objects: 100% (321/321), done.
Receiving objects:  21% (404525/1879515), 73.70 MiB | 22.00 KiB/
```

- 完成后可搜索一个三方库检验是否成功

```
pod search masonry
```

- cocoapods具体在工程中的使用

  在xcode工程目录下，创建Podfile文件：

```
pod init
```

  在文件目录生成一个Podfile文件，例如要添加masonry,Realm库,在文件中编写如下代码

```
#platform :ios, '9.0'

target 'projectname' do
    pod 'Masonry', '~> 1.1.0'
    pod 'Realm',    '3.3.0'
end
```

> 更新cocoapods版本：`gem install cocapods`
>
> 更新本地库：`pod repo update`

2.2 版本控制系统git的使用

- git简介

git是目前世界上那个最先进的版本控制系统，git是分布式的版本控制系统，文件不需要保存再中央服务器，这就意味着你完全可以在本地使用git带来的便利。git可以记录文件的每次改动，并可以让你随时会退到任何版本。不仅仅是这样，git的更大便利体现在多人协作上，大家可以统一git仓库上进行开发，大家在本地进行开发，然后将更新提交到远程仓库中，远程仓库默认只有一个master分支，而master分支的文件相对比较重要，如果提交导致了master的损坏，后果将不堪设想，因此我们在工作中常常使用如下方法使用git：

1. 将共有仓库(以learngit为例)fork到个人的仓库中-在GitHub的网站上点击fork即可
2. 将本地的ssh key添加到个人远程仓库中
3. 将个人远程仓库克隆到本地 `git clone https://github.com/Vin98/learngit`
4. 将远程仓库与本地仓库关联 `git remote add upstream https://github.com/Vin98/learngit`
5. 使用 `git remote -v` 可查看当前关联的远程库
6. 拉取远程仓库中有的本地没有的文件 `git fetch upstream`
7. 假设共有仓库上有一个分支dev_1.0，切换到这个分支 `git checkout upstream/dev_1.0`
8. 创建自己的分支 `git checkout -b lijiale_dev_1.0`
9. 更新pod库 `pod update`
10. 提交文件修改时 `git add .` 将所有更新保存到暂存区， `git commit -m "更新的内容"` 将暂存区中的文件提交到本地仓库
11. 拉取远程有的而本地没有的文件，此步骤主要是保证本地为最新文件，因为别人也有可能进行的提交 `git fetch upstream` ，将拉取到的新文件与本地文件合并 `git rebase upstream/dev_1.0` ,若成功则提交文件 `git push origin lijiale_dev_1.0` ，若不成功则多为你的更新与别人的更新发生了冲突，Xcode中会用 `c` 标识出冲突的文件，解决冲突之后再 `git add .` 然后 `git rebase --continue` ，再进行提交即可

> git中常用的一些操作:
>
> 初始化仓库 `git init`
> 查看当前仓库状态 `git status`
> 查看具体修改内容 `git diff filename`
> 回退到上一个版本： `git reset --hard HEAD^` 或者使用具体的commit-id
> `git reset --hard commit_id`
> 查看工作区和版本库最新版本的区别 `git diff HEAD -- filename`
> 丢弃工作区的修改： `git checkout -- filename` 即将filename文件在工作区的修 改全部撤销，当没有被放到暂存区时回退到版本库中的版本,当放到暂存区后会回退到暂存 区中的版本
> 撤销暂存区的修改，将暂存区的修改回退到工作区 `git reset HEAD filename`
> 当在工作区删除文件时（假设文件已在版本库中）
> 第一种情况：将文件从版本库中也删除 `git rm/add filename` `git commit -m "log"`
> 第二种情况：恢复误删的文件也就是撤销工作区的修改 `git checkout -- filename`