

Table of Contents

Articles

[快速开始](#)

[演练](#)

[进阶](#)

Blog

[GIT 的merge、rebase和cherry-pick](#)

[Google Fonts注意事项](#)

[issue trans Problem Description](#)

[Linux笔记](#)

[python 包（package）和模块（module）的创建和引入（import）](#)

[unix bsd linux shell bash GNU之间的联系，歪讲Linux\(一\)](#)

[vscode git 无需命令行](#)

[一](#)

[关于若干问题的解释说明](#)

[商品上架格式](#)

[如何在印刷品中使用遵循SIL Open Font License协议的字体](#)

[对微信支付文档的自我理解](#)

[我的python加密方案](#)

[找人](#)

[日语 时间的量](#)

[日语学习资源的分享](#)

[说明](#)

[贝多芬小传](#)

[还没有学会告别，就已经后会无期。](#)

[雷电接口](#)

[青岛科技大学新生报考参考](#)

API 文档

[ConsoleApp1](#)

[BookA](#)

[DefaultFun](#)

[JsonDocumentExtensions](#)

[StringExtensions](#)

[TestA](#)

WTuo

快速开始

这里讲解基本的知识

演练：通过一个项目来了解工作过程



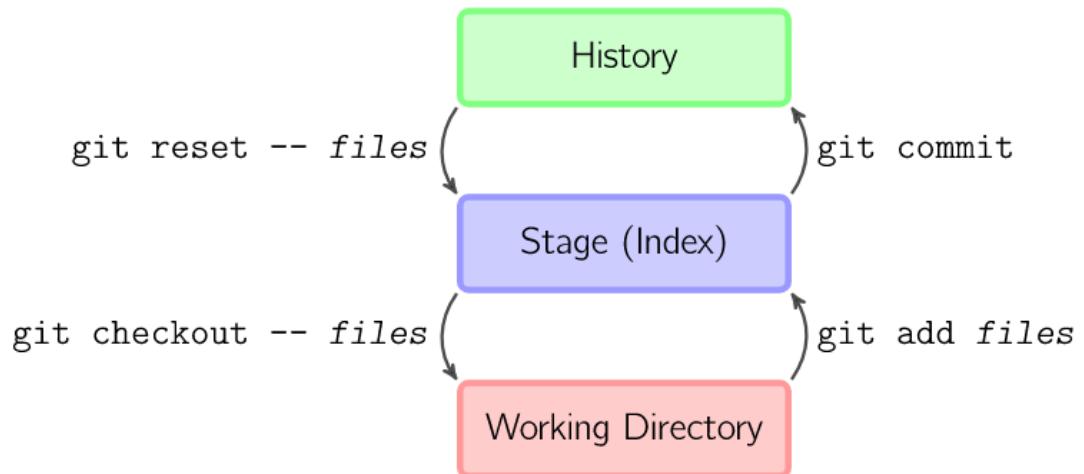
更加丰富的用法

GIT 的merge、rebase和cherry-pick区别和使用示例



名词解释

就是大体说一下git的上传和撤销的工作流程，用[图解Git \(marklodato.github.io\)](http://marklodato.github.io/visual-git-guide/index-en.html)的一张图就能说的很明白了



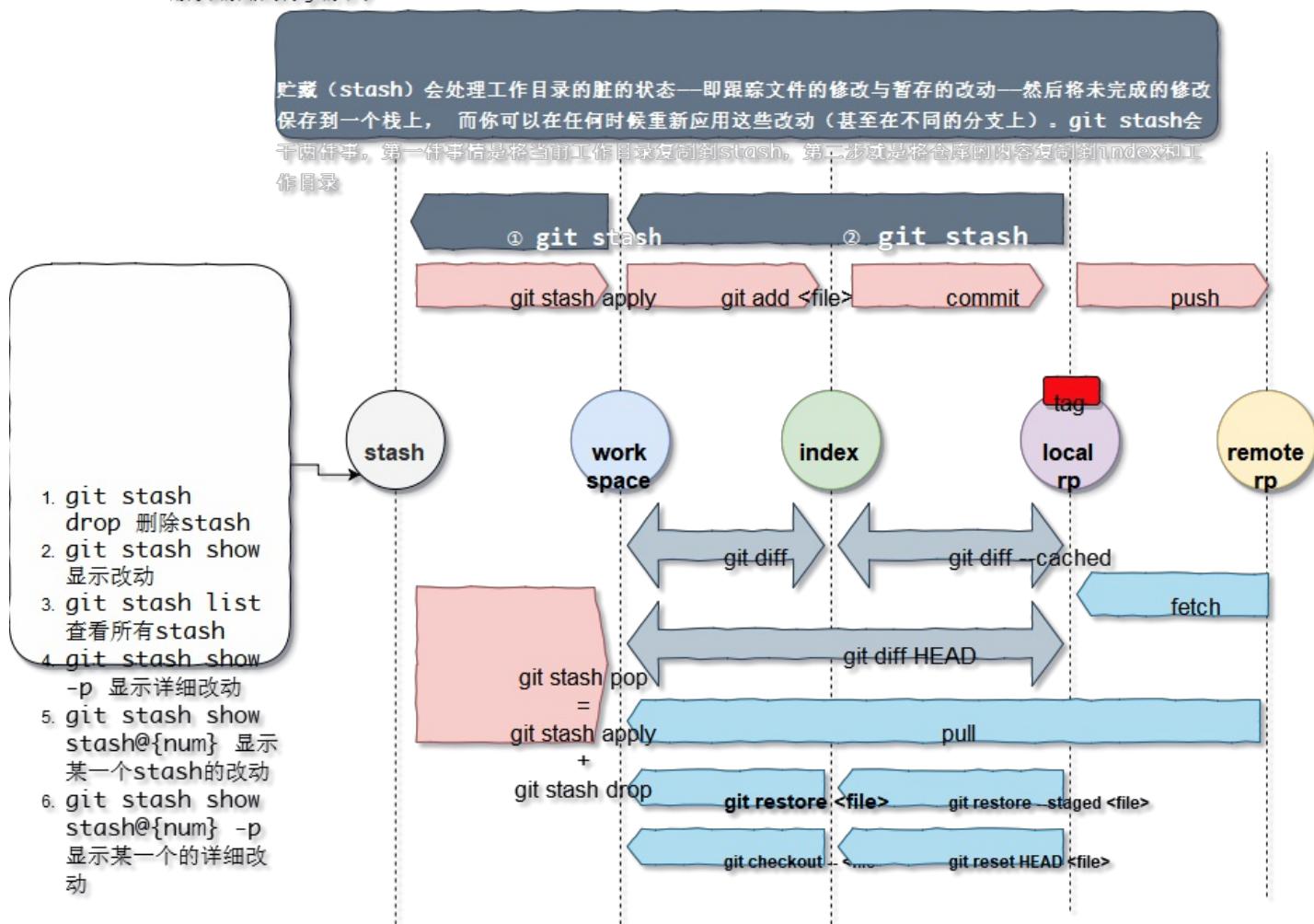
上面的四条命令在工作目录、暂存目录(也叫做索引)和仓库之间复制文件。

- `git add files` 把当前文件放入暂存区域。
- `git commit` 给暂存区域生成快照并提交。
- `git reset -- files` 用来撤销最后一次`git add files`，你也可以用`git reset` 撤销所有暂存区域文件。
- `git checkout -- files` 把文件从暂存区域复制到工作目录，用来丢弃本地修改。

或者看这张图

图解git

用简单的图来表示git的命令



当前环境

有两个分支一个是 master，另一个是 dev，两个分支都指向同一个提交，并且两个分支的状态都是干净的。

```
>git status
On branch master
nothing to commit, working tree clean
```

```
>git switch dev
Switched to branch 'dev'
```

```
>git status
On branch dev
nothing to commit, working tree clean
```

```
>git log
commit a1fe48e0054054c026554f5ed527e886113b5718 (HEAD -> dev, master)
Author: suyuesheng <df>
Date:   Sun Oct 3 16:29:12 2021 +0800

    add diff.txt
```

尝试使用 merge

① 分别修改两个分支，并提交

现在将两个分支的 `hello.pyx` 各增加一句不同的话，并提交修改，现在分支状况如下：

```
>git log --all --graph --pretty=oneline
* 42a54922a38300eb091a714f092136c469718c82 (HEAD -> master) diff master one
| * 1e04ba0525d896e46ea6d01bbbd965c50d13d03a (dev) diff dev one
|/
* a1fe48e0054054c026554f5ed527e886113b5718 add diff.txt
```

②进行合并，合并失败，需处理冲突

现在尝试合并，因为两个分支各自都分别有新的提交，所谓合并就是合并该分支基于共同祖先的更改，因为合并`dev`分支之后会伤害到`master`分支所作的更改，所以会出现如下报错

```
>git merge dev
Auto-merging cythontest/hello.pyx
CONFLICT (content): Merge conflict in cythontest/hello.pyx
Automatic merge failed; fix conflicts and then commit the result.
```

工作目录中 `hello.pyx` 中也自动标记出了[合并的冲突](#)

```
<<<<< HEAD
    print("diff master one")
=====
    print("diff dev one")
>>>>> dev
```

③处理冲突

尝试使用git自带的 `mergetool` 来处理冲突，当然也可以在工作目录中之间编辑冲突。

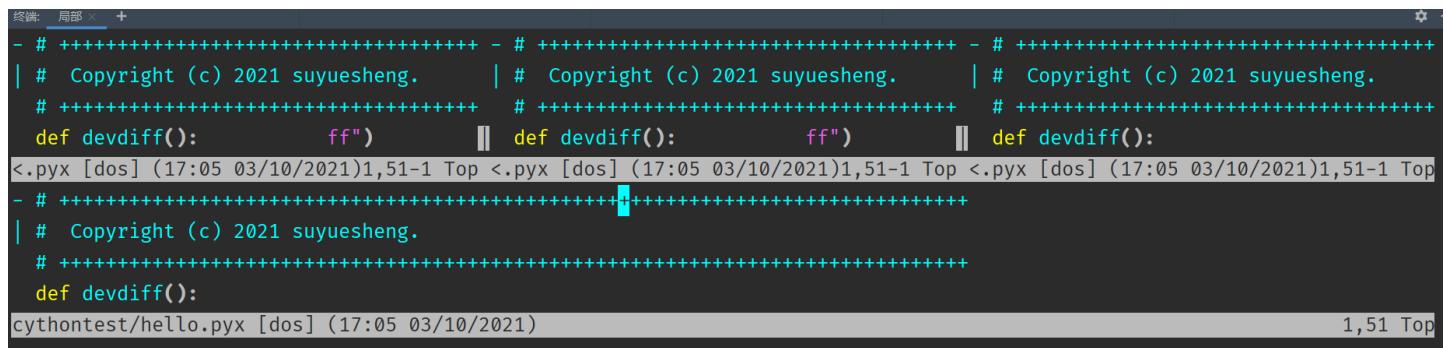
键入 `git mergetool`

```
>git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff
Merging:
cythontest/hello.pyx

Normal merge conflict for 'cythontest/hello.pyx':
{local}: modified file
{remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

键入 `回车`



The screenshot shows a terminal window in PyCharm with the following text:

```
终端: 局部 + 
- # Copyright (c) 2021 suyuesheng. | # Copyright (c) 2021 suyuesheng. | # Copyright (c) 2021 suyuesheng.
| # Copyright (c) 2021 suyuesheng. | # Copyright (c) 2021 suyuesheng. | # Copyright (c) 2021 suyuesheng.
| def devdiff():          ff")      || def devdiff():          ff")      || def devdiff():
<.pyx [dos] (17:05 03/10/2021)1,51-1 Top <.pyx [dos] (17:05 03/10/2021)1,51-1 Top <.pyx [dos] (17:05 03/10/2021)1,51-1 Top
- # Copyright (c) 2021 suyuesheng.
| # Copyright (c) 2021 suyuesheng.
| def devdiff():
cythontest/hello.pyx [dos] (17:05 03/10/2021)1,51-1 Top
```

很明显，PyCharm的终端抽风了，但是，这不重要。

.....

我退出了git自带的mergetool——`vimdiff`,我在工作目录中修改冲突。修改如下

```
print("diff master one")
print("diff dev one")
```

还记得刚键入`git merge dev`时工作目录中文件是什么鬼样子吗？忘了的话，[回去看一下](#)

现在合并工作还没有完成，因为修改冲突之后的文件还没有提交，我们先看一下现在的状态，键入`git status`

```
>git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   cythontest/hello.pyx

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   cythontest/hello.pyx
```

git非常智能地告诉我们，现在还在合并中(`still merging`)，所以就按git提示地来键入`git commit`（先键入`git add .`，将处理冲突的结果加入暂存区）来结束合并（`conclude merge`）。shell出现如下回复：

```
>git commit
hint: Waiting for your editor to close the file...
[main 2021-10-03T09:24:32.241Z] update#setState idle
[master f45308b] Merge branch 'dev'
```

④合并完成

键入`git commit`之后会自动打开一个文件，然后按git的智能提示来关闭这个文件（`close the file`），合并完成。

键入`git log --pretty=oneline --graph --all`看一下现在各个分支的状态

```
>git log --pretty=oneline --graph --all
*   f45308b77fc578dd17bcc7d7402285dca97bac0 (HEAD -> master) Merge branch 'dev'
|\ \
| * 1e04ba0525d896e46ea6d01bbbd965c50d13d03a (dev) diff dev one
* | 42a54922a38300eb091a714f092136c469718c82 diff master one
|/
* a1fe48e0054054c026554f5ed527e886113b5718 add diff.txt
```

可以看到合并成功，并且合并成功的修改被自动提交，提交信息为`Merge branch 'dev'`，当然，如果在处理冲突完之后使用`git commit -am "message"`也是可以完成合并的，只是提交信息是自定义的`message`而不是`Merge branch 'dev'`，其实`Merge branch 'dev'`就蛮好的，清楚明了。

理解

合并分支，只要两个分支各自都分别有新的提交，那么出现冲突是常态。更关键的是，如果两个分支对同一个文件进行大幅度的不同的修改，那么合并分支是一件非常蛋疼的事情，所以建议，团队开发项目的时候，尽量避免两个人同时修改同一个文件的同一部分，严格按照面向对象编程，不同的人负责不同的功能。同时推荐使用图形化合并分支工具[Git diffmerge](#)，生活好难，使用gui工具不丢人。

尝试使用cherry-pick

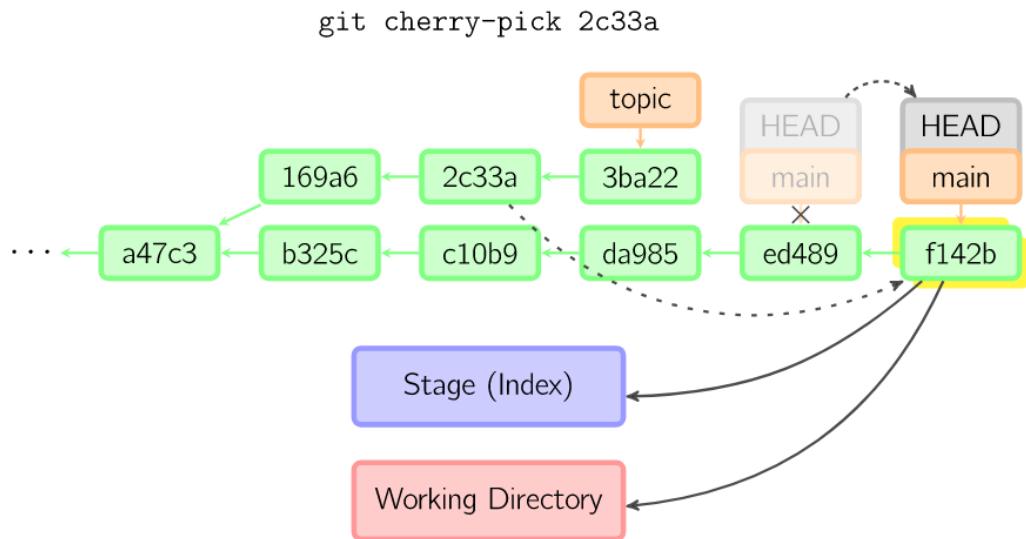
git cherry-pick 应用一些现有提交引入的更改

给定一个或多个现有提交，应用每个提交的更改，为每个提交记录一个新提交。这要求你的工作树是干净的（没有来自 HEAD 提交的修改）。

具体可以看下面的图了解 `cherry-pick`

Cherry Pick

`cherry-pick` 命令“复制”一个提交节点并在当前分支做一次完全一样的新提交。



① 当前环境

使用 `git log --all --graph --pretty=oneline` 查看两个分支的状态

```
>git log --all --graph --pretty=oneline
* c183b66776c6fdf25e9fde9b6f96d84fc0bb4c1 (HEAD -> dev)      deleted:    main.py
* de561c32bd816415a8451e3360775d2c8f97b740 diff_dev_newfile_two
| * d4c928527c7c7e16b53f918cc7a4834ec223e25f (master)  modified:   diff.txt
| * b4bcbfdc058fb74e5780755b4deebe7e743c6f5a  deleted:   diff_master_newfile_two.txt
| * 1c836275343b27b85c3809f93f7c948c58c27b8d diff_master_newfile_two
|/
*   c4992fed883bc3c2fc04694e84f892972a37a050 Merge branch 'dev'
| \
| * 1e04ba0525d896e46ea6d01bbbd965c50d13d03a diff dev one
* | 42a54922a38300eb091a714f092136c469718c82 diff master one
|/
* a1fe48e0054054c026554f5ed527e886113b5718 add diff.txt
```

在进行了 [尝试使用merge](#) 之后，两个分支各自进行了若干次提交，`master` 分支在几次提交之中，修改了 `diff.txt`。`dev` 分支增加了 `diff_dev_newfile_two.txt`，删除了 `main.py`。具体如下：

```
>git diff master --stat
diff.txt          | 14 +-----
diff_dev_newfile_two.txt |  0
main.py           | 29 -----
3 files changed, 2 insertions(+), 41 deletions(-)
```

可以说目前两个分支差距比较大了，如果合并分支的话，必定有冲突。

②进行 cherry-pick

使用 `git cherry-pick master`

```
>git cherry-pick master  
[dev 29fc0b2] modified: diff.txt  
Date: Sun Oct 3 18:34:53 2021 +0800  
1 file changed, 12 insertions(+), 2 deletions(-)
```

③ cherry-pick 完毕

看到没有，没有出现任何的冲突，并且明明是有三个文件不一样，却只是一个文件改变了，这是为什么呢？因为 `cherry-pick` 是“复制”一个提交节点并在当前分支做一次完全一样的新提交，也就是说它只是复制了 `master` 分支最近的一次提交，那么我们 `cherry-pick` 的这次提交到底修改了那些部分呢？我们比较一下 `master` 最近两次提交的hash值，

`git diff d4c9285 b4bcbfd --stat` 结果如下：

```
>git diff d4c9285 b4bcbfd --stat  
diff.txt | 14 +-----  
1 file changed, 2 insertions(+), 12 deletions(-)
```

可以看到我们 `cherry-pick` 的这次提交只修改了一个文件，所以， `cherry-pick` 之后，也只修改了一个文件。

理解

`cherry-pick` 只是复制提交，也就是说被 `cherry-pick` 的节点提交了什么，那么进行 `cherry-pick` 的节点也将会提交什么。

甚至提交的 message 也是一样的，就像我们这次 `cherry-pick` 之后，两个节点的提交信息都是 `modified: diff.txt`。

```
* 29fc0b2 (HEAD -> dev) modified: diff.txt  
* c183b66 deleted: main.py  
* de561c3 diff_dev_newfile_two  
| * d4c9285 (master) modified: diff.txt
```

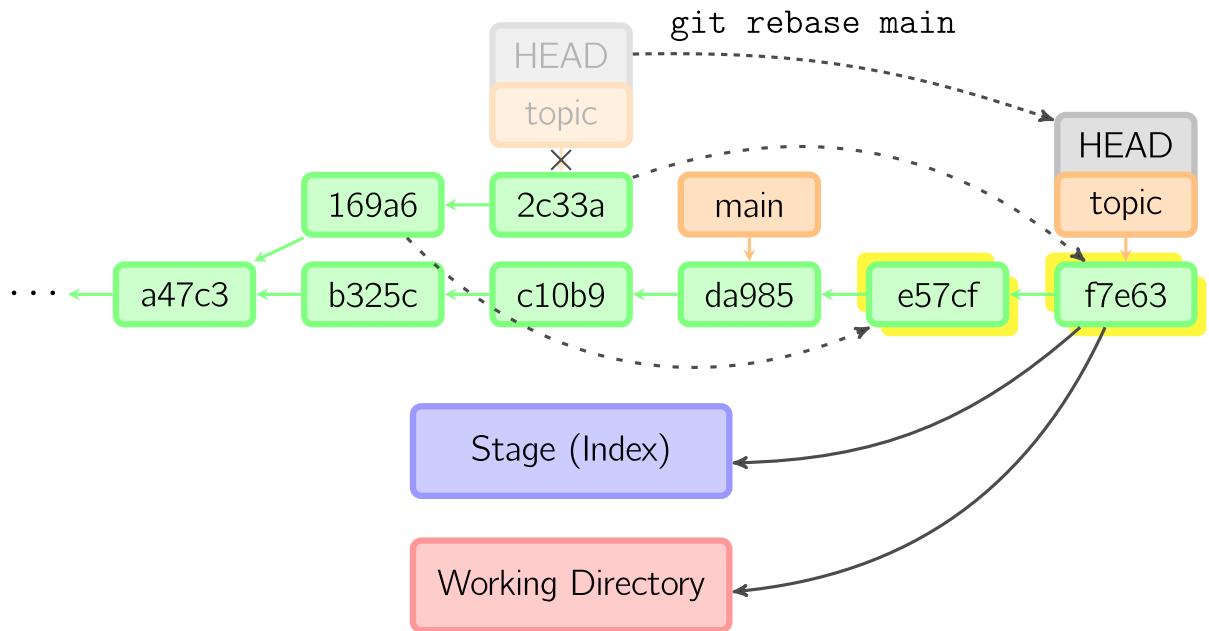
具体的，可以看下面的视频，讲解的很棒，来自 [Git cherry pick tutorial. How to use git cherry-pick. - YouTube](#)。

尝试使用 rebase

`rebase` 是合并命令的另一种选择。合并把两个父分支合并进行一次提交，提交历史不是线性的。衍合在当前分支上重演另一个分支的历史，提交历史是线性的。本质上，这是线性化的自动的 `cherry-pick`

变基的作用就是修整历史，将分支历史并入主线。

这功能看起来没啥卵用，其实这功能是需要在某些场景下才会有明显作用，比如当我们向他人维护的开源项目提交修改时，肯定要先在自己的分支中进行开发，然后再提交，但如果我们变基后再提交，维护人员就不用进行整合工作了，直接快速合并即可。



①当前环境

```
git log --graph --oneline  
* 169a6 [topic] modified: .gitignore  
* 2c33a [topic] modified: diff.txt  
* b325c [topic] deleted: main.py  
* a47c3 [topic] modified: diff.txt  
* c10b9 [topic] deleted: diff_master_newfile_two.txt  
* da985 [main] diff_master_newfile_two  
* e57cf [topic] diff_dev_newfile_two  
* f7e63 [topic] Merge branch 'dev'  
* 169a6 [topic] diff master one
```

可以看到两个分支各自都有很多新的提交。

两个分支的差别如下

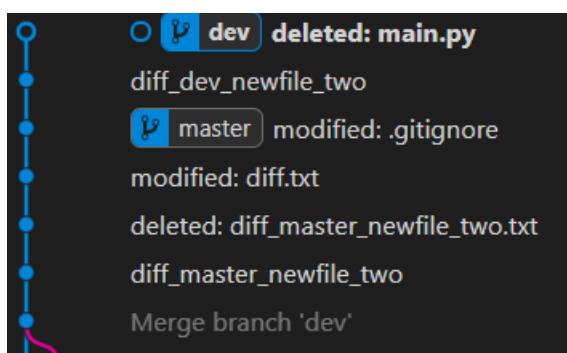
```
git diff dev --stat  
  
.gitignore | 4 ++++  
diff_dev_newfile_two.txt | 0  
main.py | 29 ++++++  
3 files changed, 32 insertions(+), 1 deletion(-)
```

可以看到修改了三个文件。

②尝试rebase

先切换到dev分支，执行`git rebase master`，它和`cherry-pick`一样都只是复制提交，而没有进行合并操作。

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: diff_dev_newfile_two
Applying:     deleted:    main.py
```



可以看到，分支变直线了，然后，切换到master之后，再合并dev。提交历史变得异常整洁。

③理解

一般我们这样做的目的是为了确保在向远程分支推送时能保持提交历史的整洁——例如向某个其他人维护的项目贡献代码时。在这种情况下，你首先在自己的分支里进行开发，当开发完成时你需要先将你的代码变基到 `origin/master` 上，然后再向主项目提交修改。这样的话，该项目的维护者就不再需要进行整合工作，只需要快进合并便可。

这几种不同的所谓的操作分支的方式有啥意义？

我认为，我们学的是功能，而不是学的“意义”，“意义”的意思是使用场景，总有一个使用场景会用到特定的功能，可能这个使用场景因为习惯的原因，我们一辈子也遇不到一次，举个极端的例子，如果从始至终就是整个仓库就一条分支，那么就肯定很难遇到合并分支的操作。如果你的团队有几百个人，每个人都负责不同的一小部分，整个仓库有上百条分支，那么变基这事就显得尤为重要，要不然就太乱了，一个芝麻大小的功能更新不配让提交历史变得杂乱。就像git官网说的这样口

至此，你已在实战中学习了变基和合并的用法，你一定会想问，到底哪种方式更好。在回答这个问题之前，让我们退后一步，想讨论一下提交历史到底意味着什么。

有一种观点认为，仓库的提交历史即是记录实际发生过什么。它是针对历史的文档，本身就有价值，不能乱改。从这个角度来看，改变提交历史是一种亵渎，你使用谎言掩盖了实际发生过的事情。如果由合并产生的提交历史是一团糟怎么办？既然事实就是如此，那么这些痕迹就应该被保留下来，让后人能够查阅。

另一种观点则正好相反，他们认为提交历史是项目过程中发生的事。没人会出版一本书的第一版草稿，软件维护手册也是需要反复修订才能方便使用。持这一观点的人会使用 `rebase` 及 `filter-branch` 等工具来编写故事，怎么方便后来的读者就怎么写。

现在，让我们回到之前的问题上来，到底合并还是变基好？希望你能明白，这并没有一个简单的答案。Git是一个非常强大的工具，它允许你对提交历史做许多事情，但每个团队、每个项目对此的需求并不相同。既然你已经分别学习了两者的方法，相信你能够根据实际情况作出明智的选择。

总的原则是，只对尚未推送或分享给别人的本地修改执行变基操作清理历史，从不对已推送至别处的提交执行变基操作，这样，你才能享受到两种方式带来的便利。

LICENSE

本文使用的部分图片来自图解Git (marklodato.github.io)，Copyright © 2010, [Mark Lodato](#). Chinese translation © 2012, [wych](#).
采用[创用CC 姓名标示-非商业性-相同方式分享3.0 美国授权条款](#)授权。本文嵌入的视频视频详细信息点击[YouTube](#)。
除文中已经表明引用的部分之外的其他部分版权来自署名-非商业性使用-相同方式共享 3.0 美国 (CC BY-NC-SA 3.0 US) © 2021 苏月晟。

[Google Fonts](#)是一个字体嵌入服务库。这包括免费和开源字体系列、用于浏览器的交互式 Web 目录以及用于通过 CSS 和 Android 使用字体的 API。Google 字体库中的流行字体包括 Roboto、Open Sans、Lato、Oswald、Montserrat、Source Sans Pro 和 Raleway。根据官网文档，[Google Fonts](#)的字体可以在您的产品和项目中自由使用这些字体，比如印刷或数字、商业或其他方式。[Google Fonts](#)也根据人们的疑问设置了[Frequently Asked Questions](#)，但是因为这些文档都是用英语书写且有很多专业术语，理解起来十分困难。笔者仔细阅读了[Google Fonts](#)旗下字体所使用的各种版权许可证书，以Q&A的方式来解释使用[Google Fonts](#)的注意事项。希望能够帮助到使用[Google Fonts](#)的人。

关于[Google Fonts](#)的一切，应以[Google Fonts](#)官网为准，本文仅供参考，并非法律建议。

Q&A

1. 可以将字体下载到个人电脑使用吗？

可以。

2. 可以将字体用于印刷品吗？

可以。

3. 免费吗？

是的。

4. 可以商用吗？

可以免费使用这些字体在您的产品和项目打印或数字，商业或其他。这不是法律建议，请考虑咨询律师，并查看所有细节的完整执照。

5. 可以在软件或者网页中使用吗？

可以，但需要表明字体的版权归属（License文件）。

6. 可以将字体放在服务器或者分发吗？

可以，在分发的时候需要揭露字体软件版权归属。根据所使用的开源协议不同，不一定允许你将该字体售卖。

7. 可以修改字体并分发修改版本吗？

可以，在分发的时候需要揭露原字体软件版权归属。根据所使用的开源协议不同，不一定允许你将该字体售卖。也可能要求将修改字体使用相同协议分发。

8. 使用该字体创建的文档需要沿用原字体的协议吗？

不需要。针对“必须以同样授权释出”的要求规定，并不适用于任何使用该“字型软件”创建的任何文档。

9. 印刷品使用[Google Fonts](#)会面临什么不良影响？

印刷品无需因使用了[Google Fonts](#)而包含该字体的授权条款和版权声明，但是，印刷品在电脑上的源文件，比如包含字体源文件的psd、word等，如果使用了该协议授权的字体，还是需要在分发源文件的时候声明所使用字体的版权（声明方式可以是印刷品源文件和字体版权协议放在同一个文件夹下）。如果印刷品源文件只是设置使用什么字体而没有在源文件中包含字体软件的任何部分（比如说，我设置使用宋体，但是最终显示的效果要取决于你的电脑上有没有安装这个字体。文字工作者应该很好理解这个和包含字体源文件的区别。）就不用声明字体授权。

10. 为什么有些字体在其他字体商是收费的？这些字体可以免费使用吗？

只要字体是从Google Font获得的，就可以免费使用。字体是否在外部字体商收费不影响Google Font下的所有字体是免费的事实。Search queries may surface results from external foundries, who may or may not use open source licenses.

11. 如何在使用的时候声明字体版权？

版权声明与条款全文可以被放置在独立纯文本文件、人类可读信息头、或文本 / 二进制文件内适当的、用户易于查阅浏览的机器可读元数据字段。

12. 举例说明在使用的时候声明字体版权或者如何放置字体的License?

- 在网站中使用Google Font的时候，可以将字体文件和字体版权声明放在一个字体名命名的文件夹内。如果使用多个字体，这些字体名命名的文件夹可以放在Font文件夹内。
- 在开发的app中使用Google Font的时候，可以在软件安装包里放置版权声明。也可以在软件界面的某个地方放置版权声明（比如点击License按钮之后，显示所使用开源字体的版权声明）。
- Google Font中字体软件所使用的版权声明可以在[google/fonts: Font files available from Google Fonts, and a public issue tracker for all things Google Fonts \(github.com\)](https://github.com/google/fonts)获得。
- 以上声明字体版权的方式并非唯一选项，仅供参考，不是法律要求或建议。

13. 在什么情况下不用声明字体版权？

- 建议在任何时候都去声明字体版权来表达对原作者的尊重，但是确实在某些情况下不用声明字体版权。
- 通过使用自[Browse Fonts - Google Fonts](https://fonts.google.com/)获得的代码或者字体文件是链接到Google Font放置字体软件的位置时，可以不用声明版权，因为这种方式本质上是使用的Google服务器或者Google的Github仓库上的字体，Google已经在这些地方声明了完整的版权了。
- 在分发印刷品时，可以不声明版权，因为印刷品并非软件。但是印刷品在电脑上的源文件属于软件范畴，需要声明使用字体的版权。
- 如果文档只是设置使用什么字体而没有在源文件中包含字体软件的任何部分（比如说，我设置使用思源宋体，但是最终显示的效果要取决于你的电脑上有没有安装这个字体。文字工作者应该很好理解这个和包含字体源文件的区别。）就不用声明字体授权。

14. 在分发、播放和演示使用了Google Font字体的视频时需要声明字体版权吗？

必须声明版权，声明版权的方式可以在视频内部声明版权或者将字体版权声明文件放在和视频同一个文件夹中。同理，**word**、**pdf**、**ppt**等其他文本文件也必须声明版权。

15. 可以用字体名称和字体作者来宣传自己的产品吗？

不可以，除非取得版权持有者的书面同意。但是向版权持有者和作者的贡献致谢而标示姓名是被允许的。

License

copyright © 2021 [苏月晟](#)，版权所有。



本作品由苏月晟采用[知识共享署名-非商业性使用-相同方式共享 4.0 国际许可协议](#)进行许可。

Problem Description

Problem recurrence

When I use **English to Chinese translation, everything is normal**, as shown below □

```
suyuesheng@DESKTOP-OKC70BA:~$ trans en:zh hello  
hello  
/hə'lō/
```

你好
(Nǐ hǎo)

hello的定义
[English -> 简体中文]

感叹词
你好!
Hello!, Hi!, Hallo!
喂!
Hey!, Hello!

hello
你好， 您好

But **when translating Chinese into English, garbled characters will appear**, as shown below□

```
suyuesheng@DESKTOP-OKC70BA:~$ trans zh:en 好  
‰%
```

‰%

‰% 的翻译
[简体中文 -> English]

‰%
‰%, ‰, ‰

I am using windows terminal

Locales

My computer supports English and Chinese. I use the `locale -a` command to clearly see the following

```
C  
C.UTF-8  
en_US.utf8  
POSIX  
zh_CN.utf8  
zh_HK.utf8  
zh_SG.utf8  
zh_TW.utf8
```

The locale of my computer is Chinese, when I use the `echo $LANG` command, the following□

```
zh_CN.utf8
```

I am using a font `fira code`

My version information

The linux on my computer is ubuntu linux kernel version is #1 SMP Wed Oct 28 23:40:43 UTC 2020 | trans -v is □

```
Translate Shell      0.9.6.6

platform          Linux
gawk (GNU Awk)    4.1.4
fribidi (GNU FriBidi) 0.19.7
audio player       [NOT INSTALLED]
terminal pager     less
terminal type      xterm-256color
user locale        zh_CN.utf8 (Chinese Simplified)
home language      zh-CN
source language    auto
target language    zh-CN
translation engine google
proxy              [NONE]
user-agent         Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/602.1 (KHTML, like Gecko) Version/8.0
Safari/602.1 Epiphany/3.18.2
theme              default
init file          [NONE]

Report bugs to:   https://github.com/soimort/translate-shell/issues
```

In order to make it easier for native Chinese speakers to clearly understand the issue I provided, I will attach the Chinese information here.

使用 **wsl2**, Linux发行版是Ubuntu。

shell的语言环境是中文, `locale -a` 包含中文和英文

可以正常的从英文翻译成中文, 但是一旦用中文翻译成英文就会乱码。就像下面这样□

```
suyuesheng@DESKTOP-OKC70BA:~$ trans zh:en 好
好

好 的翻译
[ 简体中文 -> English ]

好
好, 好
```

笔者序言

2021年9月25日

2021年9月24日, 对过去写的一些关于Linux的博客尝试做了一个整合的工作, 我发现这个工作比我想像的要复杂, 因为对之前写的文章多有不满, 于是我尽力让文章变得通俗易懂且将所有引用的地方都标注出来, 在整理的过程中, 我有了一个大胆的想法, 我想让这篇文章最终变成一个实用的Linux入门文章(或者说是书籍), 这个工作不是一天就能完成的....., 但是, 这个工作毕竟已经开始了。

我主要参考的书籍是《鸟哥的Linux私房菜》, 这是一本很优秀的书籍, 我个人感觉这本书有一个缺点就是全书读下来, 缺乏连贯性, 这本书感觉像一本“特别幽默的字典”, 我很喜欢这本书, 这本书在官网是可以免费阅读的, 我其实是有用这篇文章致敬“鸟哥”的想法, 但我的文章的语言一定不会是幽默的, 我的语言可能会偏口语一些。

编写这篇文章将会是一个不太漫长的过程(一个月左右), 我会及时在我的博客[sogeisetsu - 博客园\(cnblogs.com\)](#)上更新文章, 预计两天一更吧, 每次更新我会在笔者序言里面进行简要说明更新的内容, 每次更新都会在笔者序言里拥有一个二级标题, 二级标题题目是我上传更新的日期, 按从后到前排序, 也就是说最新的更新说明会排在笔者序言最前面, 然后我应该不会按既有的目录进行更新。

这是第一次更新, 算是这个工作的开头, 更新内容如下

1. 编写目录。
2. 增加[历史与常用关键词](#)一节。
3. 在[服务管理](#)一节, 增加了三个表格。
4. 其他某几个小节, 零零散散写了一些东西。

更新于2021年9月25日02点53分凌晨

历史与常用关键词

名词解释

下面的东西, 看了就会晕, 我也是因为晕, 才写了这篇文章

- **unix** : Unix是20世纪70年代初出现的一个操作系统, 闭源, 需要付费使用。
 - **BSD** : 又被称作是伯克利软件发行版, 是**unix**的一个重要的分支, 它创造性地加入了vi(一个编辑软件)和csh(一款shell)。
 - **POSIX标准**: 一个标准, 因为**unix**的分支越来越多, **POSIX**标准的目的是实现**UNIX**的标准化, 这个标准的一个比较大的贡献就是**shell**脚本在大多数情况下, 既可以在**linux**上运行, 也可以在**mac os**上运行。
- **GNU** : 全拼是**GNU is Not Unix** 是一个[自由软件集体协作计划](#), 1983年9月27日由[理查德·斯托曼](#)在[麻省理工学院](#)公开发起。它的目标是创建一套完全[自由的操作系统](#), 称为**GNU**。理查德·斯托曼最早在[net.unix-wizards](#)新闻组上公布该消息, 并附带一份《[GNU宣言](#)》等解释为何发起该计划的文章, 其中一个理由就是要“重现当年软件界合作互助的团结精神”。
- **Linux**: 有两个说法下面
 - **操作系统内核**: Linus发布的遵循**GNU**旗下“通用公共许可证”的操作系统内核。但是, **Linux内核并非GNU计划的一部分**。^[^7]
 - **操作系统**: [GNU官网](#)更喜欢称之为**GNU/Linux**, 因为**GNU**提供了**bash**和**GCC**等操作系统所必须的东西, **Linux**操作系统包涵了**Linux内核**与其他自由软件项目中的**GNU**组件和软件, 甚至“**linux操作系统**”之中**GNU**计划的代码所占的比重要远远高于**Linux内核**, 现在所谓的各种**Linux**发行版其实应该叫做一个带有**Linux**的**GNU**系统。现在越来越多的**Linux**发行版也认为自己是“**GNU/Linux**”。
- **Linux发行版**: 有两种说法, 一种是使用了**Linux内核**就叫**Linux发行版**, 一种是只有**GNU/linux**才可以叫**Linux发行版**, **Linux发行版**之间的区别就是预装软件、软件包管理、驱动、内核补丁、桌面环境的不同。

- Linux发行版有社区版和商业版之分，前者是社区维护，后者嘛，就是公司维护咯。
- 之所以会有发行版的区别，主要还是侧重点不一样，目标用户不一样，风格不一样，背后的哲学思考不一样。
 - **Ubuntu**注重的是功能齐全，桌面华丽，预装驱动全，基本上安装了就能用口。
 - **debain**比较注重开源精神，系统里面就很少有预装闭源的软件或者驱动为了追求稳定性，软件包也不是很新
 - **Redhat**就是注重服务器系统的稳定性了，各种对kernel的补丁保证了稳定性，付费使用。
 - **centos**就是Redhat的每半年复制版（剔除了些许Redhat专有代码），继承了Redhat稳定性的特点，又由社区维护，免费用（但是**2021年之后centos8就停止维护了**，就挺震惊的口♀口）。
 - **Fedora**由红帽赞助，红帽各种新功能会先扔到Fedora上测试，算是红帽的先期测试版。
 - **Deepin**就比较牛逼了，基于debain，桌面环境用起来比较舒服，为国人的使用习惯做了优化，预装的软件很良心，自带软件商店且软件商店丰富度尚可，自带图形化包安装器，最新版甚至可以直接安装安卓软件，极大限度的照顾了小白用户，但就笔者使用下来的使用体验来看，照windows这种专业的图形界面操作系统还有相当长的差距。
 - **arch**系统就是两个字简洁，三个字，个性化，当然了，滚动更新还更新的特快口，稳定性就差点儿事儿了口。
 - 还有一些很轻量化的系统，比如安装在树莓派上的树莓派专属**debain**就可以装在一个8g的内存卡上使用口♂口，关键是还带桌面。**TinyCore Linux**就不到20MB，竟然还有图形化桌面环境，且kernel放在内存中运行，惊人不口？
 - **Android**，这个比较有争议，争议的核心是**Linux**发行版的定义问题，Android使用了Linux的内核，但是Android并没有使用GNU套件，至少**Android绝对不是GNU/Linux**。
- 不同的包管理系统：采用不同的包管理系统，各发行版的维护者对系统的追求不一样，比如准求稳定的发行版的仓库的软件通常是稳定版，追求新技术的发行版的仓库的软件通常是比较新的版本，不同的包管理系统有不同的使用方法和社区用户。
 - **什么是包管理系统**：包管理系统的作用是方便安装、升级和自行解决依赖，「仓库」有助于确保代码已经在你使用的系统上进行了审核，并由软件开发者或包维护者进行管理。
 - **包管理系统的区别**：有的包管理系统是图形化界面（GUI），有的包管理系统是命令行界面（CLI），命令行界面虽然使用方式不同，但笔者在使用体验上没有感觉到明显的不同口♂口，**apt**和**yum**都能很好的解决依赖问题，也都能方便的去安装、升级和卸载包。据说不同的包管理系统在处理依赖的能力上是有区别的。
 - **为什么包管理系统没有统一**：主要是历史原因。大约在同一时间建立了多个软件包管理系统，特别是**.rpm**和**.deb**。每个都有自己的拥护者，每个都足够优秀口，以至于没有任何一个软件包管理者具有明显的优势。发行商肯定不会完全重建他们的系统以实施其他程序包管理器。比如笔者我就比较喜欢**yum**，因为我单纯觉得**yum**的命令比较好记和简洁，**apt**虽然使用起来比较贴合直觉，但早些年的**dpkg**在安装本地的包的时候竟然不会解决依赖问题，需要用**gdebi**来解决口（当然，新版的**debain**可以用**apt-get**来安装本地包并且解决依赖了口）
 - **主要的包管理系统使用方式的区别**：这个随便一搜就出来了，贴个随便搜的链接[Linux 包管理基础: apt、yum、dnf 和 pkg - 知乎 \(zhihu.com\)](#)
 - **遇到仓库没有的软件怎么办**：一是自己编译源代码，比如用**make**命令编译，二是添加源，添加源，添加的源里面就收入了这个软件。贴一个常用的清华源（注意：这是镜像源，理论上来说，原版源没有的软件，清华的镜像源里面也没有）[centos | 镜像站使用帮助 | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror](#)

● **Android:**

- 名词解释（绝大多数人撕逼的地方就在这里）：安卓有很多含义
 - **AOSP**：Android Open Source Project Android 开放源代码项目，遵循阿帕奇协议，谷歌是该项目的维护者。
 - 商标：谷歌公司注册了，属于谷歌公司。
 - **Android**：这个安卓指的是谷歌公司发布的包含GMS服务和各种谷歌软件和新特性的操作系统，Android=AOSP+GMS+新特性。
- 开源还是闭源：
 - Android是运行于[Linux kernel](#)之上，但并不是[GNU/Linux](#)。
 - Android默认情况下并没有本机[X窗口系统](#)，也不支持整套标准[GNU](#)库。Android为了达到商业应用，必须移除被[GNU GPL](#)授权证所约束的部分。

- 只有基础的**Android**操作系统（包括一些应用程序）才是开源软件，任何厂商都不须经过Google和开放手持设备联盟的授权随意使用Android操作系统；大多数Android设备都附带着大量的专有软件，例如是[Google移动服务](#)，当中包括[Google Play](#)商店、[Google](#)搜索，以及[Google Play服务](#) — 那是一个提供与Google提供的服务[应用程序接口](#)集成的软件层。这些应用程序必须由设备制造商从Google得到许可
 - Android的硬件抽象层是能以封闭源码形式提供硬件驱动模块。HAL的目的是为了把Android framework与Linux kernel隔开，让Android不至过度依赖Linux kernel，以达成“内核独立”（kernel independent）的概念。这样安卓剥离了Linux内核以外的东西，只需要对**Android**修改过的Linux内核采用**GPL**协议，硬件驱动可以闭源，**Android**其余的可以采用阿帕奇协议，保护了硬件厂商的权益和安卓**GMS**和安卓其他新特性，这也就是为什么华为可以使用**ASOP**却不能使用**GMS**和**Android12**。
- **shell**: 壳程序，通俗的讲是命令解释器，他的作用就是将用户的指令告诉操作系统的内核（**kernel**），然后呢**kernel**调用电脑的各种软硬件来达到你的指令的要求。Windows的cmd是shell，Bash是Shell，powershell是shell，甚至windows的文件资源管理器也是shell（图形界面shell），sh也是shell（地球上第一个比较流行的shell简称sh，是一个外国人开发出来的，简称Bourne shell）
 - 分为图形界面**shell**和文本**shell**，当然，这个区分是我瞎编的，如有雷同，不胜荣幸哦口
 - **bash** : 地球上所有能叫出名字来的Linux发行版默认使用的shell，可以说bash是sh的宇宙无敌增强版，知乎上竟然有一个问题linux初学者是学csh还是tcsh还是sh还是zsh的，我晕，初学者当然是学**bash**啊，虽然是大同小异，可实在是没有必要让一个初学者第一个学习的**shell**竟然不是**bash**。其实吧，问这个问题的同学，你甚至可以学习在linux上用powershell，powershell是完全有能力向**kernel**解释你的指令的口，当然啦，在linux上使用powershell是一件可以但没有必要的事情。
 - 不同**shell**的区别？首先是功能上的区别，有的**shell**的自定义程度很高，可以自定义彩色显示，有的**shell**可以自动建议和自动补全命令，有的**shell**会匹配历史命令，这是功能上的区别。然后就是标准的区别，也就是是否支持**POSIX**标准，比如说fish的早期版本不支持这个标准，fish无法执行含有`&&`的命令，这对使用者造成了很大的困扰。

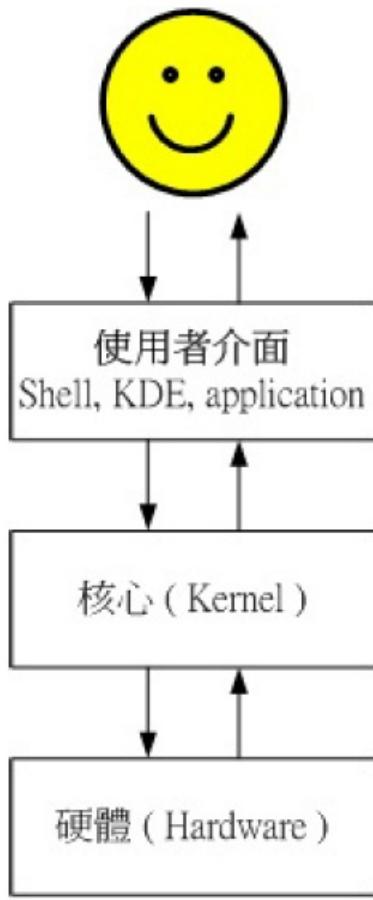
历史

（一）先从UNIX到GNU讲起

这部分很简单，20世纪六十年代，贝尔实验室，开发出了unix。**Unix**不是开源软件，**Unix**源码可以与它的拥有者**AT&T**通过协议获得许可证。第一个已知的软件许可证在1975年卖给了伊利诺伊大学。

Unix在学术界发展迅速，随着伯克利成为重要的活动中心，在70年代，Ken Thompson（开发unix的领导者）有一个学术休假。通过在伯克利的Unix的所有活动，一个新的Unix软件之父诞生了：伯克利软件发行版，或者叫**BSD**。

有好多公司和团体吧，就在unix的基础上二次开发，整出来了好多unix的分支，并且吧，这些个分支所使用的**shell**还不一样，**shell**是啥呢？他的官方名称就是**命令解释器**或者叫壳程序，他的功能就是把用户输入的命令告诉操作系统的内核，然后这个核心再指挥硬件。



您就是這個可愛的笑臉，
使用文字或圖形介面，
在螢幕之前操作你的作業系統。

接受來自使用者的指令，
以與核心進行溝通。

真正在控制硬體工作的咚咚
含有 CPU 排程、記憶體管理、
磁碟輸出輸入等工作。

整個系統中的實體工作者，
包含了硬碟、顯示卡、網路卡、
CPU、記憶體等等。
沒有他，就沒有其他的咚咚啦！

shell不一样，进行相同的操作接收的命令也就不一样，也就是说，相当于这些**shell**所能听懂的人话不一样，好比**shell**是不同国家的人，有的**shell**只能听懂英文，有的**shell**只能听懂日文，这可咋办，这地球上**unix**分支这么多，甚至不同的计算机安装的分支都不一样，总不能让用户换个电脑还得重新学门语言吧，于是乎**POSIX**标准就出来了，不管你是谁，都应该能听懂同一种语言，全世界都得说中国话口（当然这只是戏虐）！于是乎，同样的命令，可以在所有支持**POSIX**标准的**shell**上运行。

然后，[理查德·斯托曼](#)，我们就暂且叫他老李头吧，老李头对**unix**很不爽，于是乎呢，他有了一个想法，那就是开发一个完整的开放源代码的类**unix**操作系统，来取代**unix**。他给自己的这个想法起了个名字叫**GNU**，名字的意思呢是“**GNU's Not Unix!**”，翻译成中文就是“老子不是**unix**，老子叫**GNU**！！口”，他这个想法一提出来就得到了广大人民群众自发的支持，毕竟大家早就看**unix**不爽了，**unix**，啥都不是！想学习学习你的源代码竟然要花钱，老子不用你了！于是大家怀着这个要将**unix**干翻的想法开始工作起来了。

老李头在接下来的几年，成立了自由软件基金会，开始雇佣人来写代码，并且发表了操作系统界的《独立宣言》——**GNU**宣言。还整了个**GPL**协议，这个协议挺**niubility**口啊，遵循这个协议的软件必须开源，并且使用遵循**GPL**协议代码的软件也必须开源，哪怕是你商用，也得开源。



这个计划吧，一开始还算顺利，向命令解释器（bash）啊，编辑器（Emacs）啊，编译器（GCC）啊啥的都进展的挺顺，可是啊，就是核心开发不出来，核心是啥呢？上面提过一嘴，核心就是能够控制硬件来执行接收到的命令的东西。这个东西可把老李头愁坏了，他是吃也吃不下，睡也睡不着，头发大把掉，胡子蹭蹭长（只是在调侃理查德的大胡子□♀□）……

（二）天降正义，Linus！

时间来到了九十年代，一个芬兰赫尔辛基大学的大学生林纳斯，我们管他叫小林吧，这个小林不声不响，开发出来了操作系统内核linux，并且这个Linux还采用了**GPL协议**，小林按当时的话来说，就是比较帅呆了，还特谦虚幽默，你看他当年放出Linux时说的那话

我在做个（自由的）操作系统（就是个兴趣爱好，我不会搞得像GNU那么大那么专业），打算让它工作在386 AT平台上。它从四月就开始酝酿了，马上就快好了。我想要那些喜欢或不喜欢minix的人的意见，因为我的系统和它有点类似（同样的文件系统的物理布局——由于实际原因——还有些其他的东西）。

我已经移植了**bash(1.08)**和**gcc(1.40)**，而且看起来奏效了。这意味着我会在几个月内得到一些实用的东西。“……”是的——它没有任何minix代码，并且它有一个多线程的fs。它不可移植（使用386任务切换等），而且它可能永远不会支持除AT硬盘之外的其他东西，因为我只有这些:-)。

“……”它基本上是用C语言写的，但是大多数人可能不会把我写的东西叫做C语言。它使用我能找到的386的每个可以想象的特性，因为它也是一个教我关于386的功能的项目。我前面提到过，它使用内存管理单元来进行分页（还没实现到对硬盘的功能）和分段。这个分段功能使得它真正的依赖于386（每个任务都有64Mb的代码和数据段——4Gb中最多64个任务。如果有人需要超过每个任务64Mb的限制，那将是个麻烦事）。“……”我的一些C语言文件（特别是mm.c）几乎用了和C一样多的汇编。“……”不像minix，我也碰巧喜欢中断，所以中断将在不试图隐藏背后的原因的情形下被处理。



林纳斯·托瓦兹，Linux内核首创者。

老李头看这个小林是越看越喜欢，小林呢觉得GNU也不错，于是呢，\$GNU+linux kernel（内核）=GNU/Linux！\$

（三）越来越受欢迎

从GNU/Linux第一个发行版Boot-root问世以来，GNU/Linux越来越受到欢迎，Marc Ewing成立Red Hat软件公司，成为最著名的Linux经销商之一，也证明了GNU/Linux在商业上被认可（GNU/Linux虽然开源，但不代表不可以商用，Red Hat是开源的，但是并不免费，Red Hat会为Red Hat的用户有偿提供技术服务）。

现在，整个GNU/Linux操作系统家族基于Linux内核部署在传统计算机平台（如个人计算机和服务器，以Linux发行版的形式）和各种嵌入式平台，如路由器、无线接入点、专用小交换机、机顶盒、FTA接收器、智能电视、数字视频录像机、网络附加存储（NAS）等。工作于平板电脑、智能手机及智能手表的Android操作系统同样通过Linux内核提供的服务完成自身功能。尽管于桌面电脑的占用率较低，基于Linux的操作系统统治了几乎从移动设备到主机的其他全部领域。截至2017年11月，世界前500台最强的超级计算机全部使用Linux。

GNU/Linux被打包成供个人计算机和服务器使用的Linux发行版，一些流行的主流Linux发行版，包括Debian（及其派生版本Ubuntu、Linux Mint）、Fedora（及其相关版本Red Hat Enterprise Linux、CentOS）和openSUSE等。Linux发行版包含Linux内核和支撑内核的实用程序和库，通常还带有大量可以满足各类需求的应用程序。

（四）命名之争，linux还是GNU/linux？

有越来越多的人管GNU/linux叫linux操作系统，老李头团队的某些人一看不对啊，这怎么行？虽然使用了Linux内核，可也不能抹杀GNU的贡献吧？GNU起初可是要打造GNU系统的啊。

支持GNU的人认为GNU提供了bash和GCC等操作系统所必须的东西，Linux操作系统包涵了Linux内核与其他自由软件项目中的GNU组件和软件，甚至“linux操作系统”之中GNU计划的代码所占的比重要远远高于Linux内核，现在所谓的各种Linux发行版其实应该叫做一个带有Linux的GNU系统。

还有些人呢，就发话了，一个操作系统最重要的是啥？不就是内核吗？我们就叫Linux怎么了？（不代表笔者看法）

“GNU/Linux”此名称是GNU计划的支持者与开发者，特别是其创立者理查德·斯托曼对于Linux操作系统的主张。由于此类操作系统使用了众多GNU程序，包含Bash（Shell程序）、库、编译器等等作为Linux内核上的系统包，理查德·斯托曼认为应该将该操作系统称为“GNU/Linux”或“GNU+Linux”较为恰当，但现今多数人仍称其为Linux。就1997年之前的Linux来看，一间CD-ROM的供应商所提供的资料显示在他们的“Linux发行版”中，GNU软件所占最大的比重，大约占全部源代码的28%，且还包括一些关键的部件，如果没有这些部件，系统就无法工作，而Linux本身占大约3%。

Linux社群中的一些成员，如埃里克·雷蒙、林纳斯·托瓦兹等人，偏好Linux的名称，认为Linux朗朗上口，短而好记，拒绝使用“GNU/Linux”作为操作系统名称。并且认为Linux并不属于GNU计划的一部分，斯托曼直到1990年代中期Linux开始流行后才要求更名。有部分Linux发行版，如Debian，采用了“GNU/Linux”的称呼。但大多数商业Linux发行版依然将操作系统称为Linux。而有些人则认为“操作系统”一词指的只是系统的内核(Kernel)，其他程序都只能算是应用软件，因而，该操作系统应叫Linux，但Linux系统包是在Linux内核的基础上加入各种GNU软件包集合而成的。

反正吧，这事儿现在也没个准确的说法，我觉得吧，人GNU是做了很大贡献的，这个肯定都承认，可是挡不住Linux这个叫法朗

朗上口写起来还省墨水啊，总之，现在（2021年8月31日）各种官方还没有个统一的说法，叫啥名字都无伤大雅，不管什么名字，都掩盖不了GNU和linux kernel的成功，也掩盖不了Linux操作系统对这个世界（尤其是开源世界）的贡献。

以上，就是Linux从无到有的历史了，通过这些历史，我们将不再对unix, GNU, Linux, Shell和bash等概念感到一筹莫展了，这就足够了。

系统安装

虚拟机

wsl

U盘安装

双系统

安装之后要做的第一件事情，换源、更新、安装常用软件和更改语系支持
常用命令

软件安装

[换源和更新](#)

文件与目录命令

加密 `tar`、`zip` 和 `unzip`

`tar`

`tar` 是一个Linux下比较常见的压缩命令，这个命令比较简单，支持多种压缩方式，缺点是压缩出来的文件会得不到某些 windows 下常用压缩工具（7-zip）很好的支持。

应用举例

```
tar -xvf foo.tar
      提取 foo.tar 文件并显示提取过程

tar -xzf foo.tar.gz
      提取用 gzip 压缩的文件 foo.tar.gz

tar -cjf foo.tar.bz2 bar/
      用 bzip 为目录 bar 创建一个叫做 foo.tar.bz2 存档

tar -xjf foo.tar.bz2 -C bar/
      把用 bzip 压缩的文件 foo.tar.bz2 提取到 bar 目录

tar -xzf foo.tar.gz blah.txt
      把文件 blah.txt 从 foo.tar.gz 中提取出来
```

注意：当压缩或提取的时候，压缩类型选项常常是不必要的，因为 `tar` 会根据文件的后缀自动选择压缩类型。

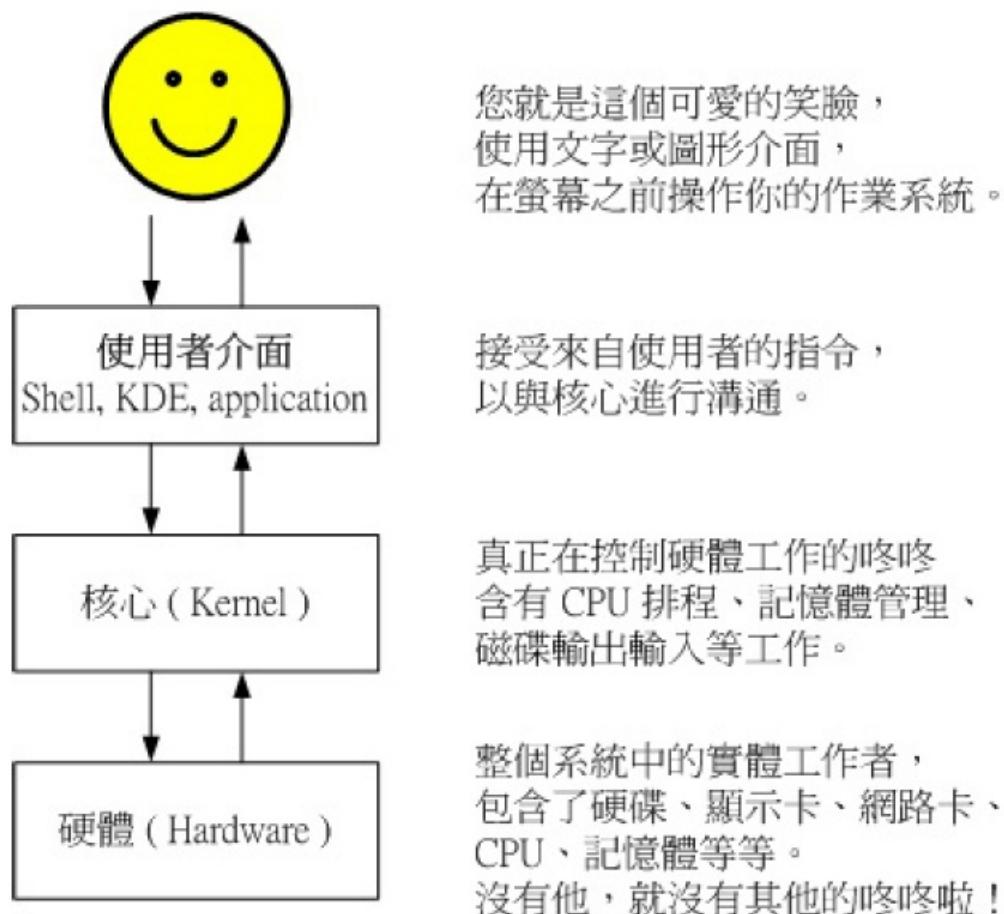
权限管理

shell 脚本

bash == 一种壳程序

shell就是“壳程序”，你可以阅读第一章的名词解释中的[shell部分](#)来了解一下。

shell通俗的讲是命令解释器，他的作用就是将用户的指令告诉操作系统的核（**kernel**），然后呢**kernel**调用电脑的各种软硬件来达到你的指令的要求。



地球上有很多shell，Linux上也预装了很多shell（根据Linux发行版的不同，预装的软件也会不同），**bash**是Linux默认的shell。

描述(DESCRIPTION)

Bash 是一个与 sh 兼容的命令解释程序，可以执行从标准输入或者文件中读取的命令。 Bash 也整合了 Korn 和 C Shell (ksh 和 csh) 中的优秀特性。

Bash 的目标是成为遵循 IEEE POSIX Shell and Tools specification (IEEE Working Group 1003.2, 可移植操作系统规约： shell 和工具) 的实现。

变量

环境变量与自有变量

变量的读取与查看 `echo`、`env`、`set`

变量类型的转换 `export`、`declare`

基本语法

运行脚本

环境配置文件

`login shell` 和 `non-login shell`

[Linux登陆的两种状态 - 孙晨c - 博客园 \(cnblogs.com\)](#)

语系的配置

中文man手册

编辑器

程序管理

process和program

- 程序 (program) : 通常为 binary program , 放置在储存媒体中 (如硬盘、光盘、软盘、磁带等) , 为实体文件的型态存在;
- 程序 (process) : 程序被触发后, 执行者的权限与属性、程序的程序码与所需数据等都会被载入内存中, 操作系统并给予这个内存内的单元一个识别码 (PID) , 可以说, 程序就是一个正在运行中的程序。

一般所说的程序, 指的是process

子程序与父程序

先明确一个概念, Linux下所有的应用都是程序, 这句话的意思是, bash也是一个程序。我们不要将bash看作是一个终端, 其实bash是一个shell程序。也就是说, 当我们打开bash的时候, 就已经有一个bash程序在运行了, 我们在bash中输入命令 (假定输入 `ps -l`) , 这个命令会产生一个程序, 这个程序叫做bash的子程序。

每一个程序都有一个单独的pid。查看pid可以使用命令 `ps -l` , 这个命令可以查看与当前bash相关的所有程序和子程序

```
suyuesheng@DESKTOP-OKC70BA:~$ ps -l
F S   UID    PID  PPID C PRI NI ADDR SZ WCHAN TTY      TIME CMD
4 S  1000    106  105  0  80   0 -  6215 wait    pts/2   00:00:00 bash
4 S  1000    289  288  0  80   0 -  6195 wait    pts/2   00:00:00 bash
4 S  1000   6259  6258  0  80   0 -  6187 wait    pts/2   00:00:00 bash
4 S  1000   7127  7126  0  80   0 -  6194 wait    pts/2   00:00:00 bash
4 S  1000   7150  7149  0  80   0 -  6187 wait    pts/2   00:00:00 bash
4 S  1000   7266  7265  0  80   0 -  6220 wait    pts/2   00:00:00 bash
4 S  1000   7330  7329  0  80   0 -  6288 wait    pts/2   00:00:00 bash
4 S  1000   7853  7852  0  80   0 -  6286 wait    pts/2   00:00:00 bash
0 R  1000   8470  7853  0  80   0 -  7350 -      pts/2   00:00:00 ps
```

第一行有很多字段, `F`、`S`、`PID`等, `pid`代表的是当前程序的id, `ppid`代表的是当前程序的父程序的id。==可以看到第11行的父程序是第10行pid为7853的程序。==

job 工作管理

这个工作管理 (job control) 是用在 bash 环境下的, 也就是说: "当我们登陆系统取得 bash shell 之后, 在单一终端机接口下同时进行多个工作的行为管理=="。举例来说, 我们在登陆 bash 后, 想要一边复制文件、一边进行数据搜寻、一边进行编译, 还可以一边进行 vim 程序撰写!

后台运行 &

后台暂停 `ctrl+z`

`jobs` 显示工作的状态

选项：

- l 在正常信息基础上列出进程号
- n 仅列出上次通告之后改变了状态的进程
- p 仅列出进程号
- r 限制仅输出运行中的任务
- s 限制仅输出停止的任务

后台的工作拿到前台 `fg`

后台暂停的工作继续运行 `bg`

停止 `ctrl+c`、`kill`

- `ctrl+c`：结束前台的工作，例如 `ping google.com` 可以用 `ctrl+c` 来结束。
- `kill`：杀死进程，`kill` 有很多不同的信号，使用 `kill -l` 可以查看信号列表，不同的信号表示对进程不同的操作。

```
suyuesheng@DESKTOP-OKC70BA:~$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

<code>KILL</code> 的 <code>SINGLE(信号)</code>	代表的操作
1	对于daemon（守护进程）是重新读取配置，对于普通进程来说就是杀掉进程
9	强制杀掉进程
15（默认）	使用正常的步骤杀死进程

有一个惯例，通常使用 `kill -9` 是因为某些程序你真的不知道怎么通过正常手段去终止他，这才用到 `kill -9` 的！

使用 `kill` 默认的方式（`kill -15`）不一定能够杀死进程。

==但凡是知道正常终止这个进程的方式，就不要用 `kill !` ==

服务管理

Linux老版本使用 `service`、`chkconfig` 等命令来管理系统服务，在 RHEL 7/8 系统中是使用 `systemctl` 命令来管理服务的，现在将两个版本的区别列在下面，表格来自 [第1章 动手部署一台Linux操作系统 | 《Linux就该这么学》 \(linuxprobe.com\)](#)

- `systemd` 与 `System V init` 的区别以及作用 □

<code>SYSTEM V INIT</code> 运行级别	<code>SYSTEMD</code> 目标名称	<code>SYSTEMD</code> 目标作用
0	<code>poweroff.target</code>	关机
1	<code>rescue.target</code>	单用户模式

SYSTEM V INIT运行级别	SYSTEMD目标名称	SYSTEMD 目标作用
2	multi-user.target	多用户的文本界面
3	multi-user.target	多用户的文本界面
4	multi-user.target	多用户的文本界面
5	graphical.target	多用户的图形界面
6	reboot.target	重启
emergency	emergency.target	救援模式

- 服务的启动、重启、停止、重载、查看状态等常用命令

老系统命令	新系统命令	作用
<code>service foo start</code>	<code>systemctl start httpd</code>	启动服务
<code>service foo restart</code>	<code>systemctl restart httpd</code>	重启服务
<code>service foo stop</code>	<code>systemctl stop httpd</code>	停止服务
<code>service foo reload</code>	<code>systemctl reload httpd</code>	重新加载配置文件（不终止服务）
<code>service foo status</code>	<code>systemctl status httpd</code>	查看服务状态

- 服务开机启动、不启动、查看各级别下服务启动状态等常用命令

老系统命令	新系统命令	作用
<code>chkconfig foo on</code>	<code>systemctl enable httpd</code>	开机自动启动
<code>chkconfig foo off</code>	<code>systemctl disable httpd</code>	开机不自动启动
<code>chkconfig foo</code>	<code>systemctl is-enabled httpd</code>	查看特定服务是否为开机自启动
<code>chkconfig --list</code>	<code>systemctl list-unit-files --type=httpd</code>	查看各个级别下服务的启动与禁用情况

参考文档

阮一峰的博客：[阮一峰的网络日志 \(ruanyifeng.com\)](http://ruanyifeng.com)。

《鸟哥的Linux私房菜》

《Linux就该这么学》

python 包（`package`）和模块（`module`）的创建和引入（`import`）



名词解释

实际上，Python中的函数(Function)、类(Class)、模块(Module)、包库(Package)，都是为了实现模块化引用，让程序的组织更清晰有条理。

□通常，函数、变量、类存储在被称为模块(Module)的.py文件中，一组模块文件又组成了包(Package)。

□将函数、变量、类存储在独立的.py文件中，可隐藏代码实现的细节，将不同代码块重新组织，与主程序分离，简化主程序的逻辑，提高主程序的可读性。

□有了包和模块文件，可以在其他不同程序中进行复用，还可以使用其他人开发的第三方依赖库。

本引用为CSDN博主「虾米小馄饨」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。原文链接：https://blog.csdn.net/Bit_Coders/article/details/119318000

package实际上就是就是一个文件夹，里面包含诸多module和一个`__init__.py`，package是module的一种，这点在python报错的时候也能看出来。

引入方式

1. `import moduleName`
2. `import packageName`
3. `from packageName import moduleName\packageName`
4. `from moudleName import Function\Class`

引入父级目录模块

`sys.path` 是 `sys` 模块中的内置变量。它包含一个目录列表，编译器将搜索所需的模块。

如果要引入父级模块，需要在引入之前需要在python的编译器的环境变量中添加当前文件父目录，然后再import，有两个添加方法

1. `sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))`
2. `sys.path.append("..")`

建议使用第一个方法，第二个方法会在除pycharm以外的地方运行的时候造成错误，原因是`sys.path.append(..)`添加的是当前使用者所在目录的父目录，而不是当前这个文件的父目录。

`util.hi` 是父级目录中的模块，引入方式如下：

```
import sys
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import util.hi
```

注意事项

- ✓□在没有`from`的情况下，`moduleName`的形式可以是通过附属关系按照`packageName.moduleName`使用。
- □在有`from`的时候，`import`后面必须是包名称或者是函数名或者类名或者模块名。但是不能用`.`来表示层级关系，也就是说不能用向`packageName.moduleName`之类的用法，但是可以用`,`来区分不同的模块。
- ✓□只有在有`from`的情况下，`import`后面才能跟函数名或者类名

示范

这是当前的环境口

```
.  
├── main.py  
└── test  
    └── testproject  
        ├── __init__.py  
        └── pa1  
            ├── __init__.py  
            └── hello.py  
        └── testproject.py  
└── util
```

这是pa1目录下的hello.py口，有一个函数hello和一个类HelloT。

```
import sys  
  
acb: str = "1232"  
  
def hello():  
    print("hello world")  
  
class HelloT:  
    def __init__(self) -> None:  
        pass  
  
    def hello(self):  
        print("call", sys._getframe().f_code.co_name)
```

正确示范

下面示范在根目录下，main.py的正确import示范：

```
# 引入pa1包的hello.py模块  
from testproject.pa1 import hello  
# 引入pa1包  
import testproject.pa1  
# 引入hello.py模块  
import testproject.pa1.hello  
# 引入在testproject包的pa1包  
from testproject import pa1  
# 引入hello.py模块下的hello函数和HelloT类  
from testproject.pa1.hello import hello,HelloT
```

错误示范

下面示范在根目录下，main.py的错误import示范口，错误原因请对照注意事项

```
from testproject import pa1.hello  
import testproject.pa1.hello.HelloT
```

使用方式

先看一下在引入默认模块（比如os、math、random）的时候，使用被引入的模块的方式：

```
>>> import os
>>> os.path.abspath(".")
'C:\\\\Users\\\\苏月晨\\\\Desktop\\\\pythonProject1'
>>> import math
>>> math.pi
3.141592653589793
>>> import random
>>> random.random()
0.11531493534041015
```

使用引入基本上只有两个要求，一个是别重名，一个是使用引入的时候所使用的被引入模块名字必须是和 `import` 后面的一模一样，比如说使用了 `import testproject.pa1.hello`，那么想使用 `hello` 模块的时候必须用 `testproject.pa1.hello` 而不是 `hello`。如果是使用了 `from testproject.pa1 import hello` 来引入 `hello` 模块，则在使用 `hello` 模块的时候直接用 `hello`。

可以看一下下面的示例来具体了解其中的差异：

```
>>> import os.path
>>> path.abspath(".")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'path' is not defined
>>> os.path.abspath(".")
'C:\\\\Users\\\\苏月晨\\\\Desktop\\\\pythonProject1'
>>> from os import path
>>> path.abspath(".")
'C:\\\\Users\\\\苏月晨\\\\Desktop\\\\pythonProject1'
```

使用模块中的常量

只需要在模块中定义一个常量，然后在使用的时候用 `模块名.常量名` 就可以了，就像 `hello` 模块里面有一个常量 `acb`，在引入 `hello` 模块之后，用 `hello.acb` 就可以调用常量 `acb` 了

创建方式

创建包的时候，包目录里面必须有 `__init__.py`，这个文件一般情况下可以是空的，具体这个文件怎么使用可以看 [Python init.py 作用详解 - Data&Truth - 博客园 \(cnblogs.com\)](#)

看一下下面的目录结构

```
.
├── __init__.py
└── pa1
    ├── __init__.py
    └── hello.py
└── testproject.py
```

`pa1` 是一个包，`pa1` 目录下面有一个 `__init__.py`，`pa1` 下面还有一个 `hello.py`，这个文件是一个模块。

`hello.py`

```
import sys

acb: str = "1232"

def hello():
    print("hello world")

class HelloT:
    def __init__(self) -> None:
        pass

    def hello(self):
        print("call", sys._getframe().f_code.co_name)
```

这样就创建了一个模块。

LICENSE

 署名-非商业性使用-相同方式共享 3.0 美国 (CC BY-NC-SA 3.0 US) © 2021 苏月晟。

不客观，非权威

这是本人的学习笔记，不代表权威。文中相关主观看法不代表主流的客观的看法。不对文章所涉及内容的权威性和正确性做任何保证。

说实在的，我一直认为了解这些东西其实没什么卵用，学习使用linux进行开发为什么要知道什么叫**GNU**？

但还是需要了解这些，一是为了方便自己跟别人吹牛逼，二是为了听懂别人吹的牛逼。毕竟一帮人在那谈一些像freeBSD与mac os之间的关系和华为鸿蒙与安卓与linux与unix的关系的时候，如果咱连听都听不懂就尴尬了，所以，花时间了解一下还是必要的，虽然靠吹牛逼赚不了钱，但咱也不能让别人轻轻松松把逼装到是不是？

名词解释+吐槽口（新手建议跳过本章）

下面的东西，看了就会晕，我也是因为晕，才写了这篇文章口

- **unix**： Unix是20世纪70年代初出现的一个操作系统，闭源，需要付费使用。^[^1]
 - **BSD**： 又被称作是伯克利软件发行版，是**unix**的一个重要的分支，它创造性地加入了vi（一个编辑软件）和csh（一款shell）^[^2]
 - **POSIX标准**： 一个标准，因为**unix**的分支越来越多，**POSIX**标准的目的是实现**UNIX**的标准化^[^3]，这个标准的一个比较大的贡献就是**shell**脚本在大多数情况下，既可以在**linux**上运行，也可以在**mac os**上运行。^[^4]
- **GNU**： 全拼是**GNU is Not Unix** 是一个自由软件集体协作计划，1983年9月27日由理查德·斯托曼在麻省理工学院公开发起。它的目标是创建一套完全自由的操作系统，称为**GNU**。理查德·斯托曼最早在net.unix-wizards新闻组上公布该消息，并附带一份《**GNU宣言**》等解释为何发起该计划的文章，其中一个理由就是要“重现当年软件界合作互助的团结精神”。^[^5]
- **Linux**： 有两个说法下面口
 - 操作系统内核： Linus发布的遵循**GNU**旗下“通用公共许可证”的操作系统内核^[^6]。但是，**Linux**内核并非**GNU**计划的一部分。^[^7]
 - 操作系统： **GNU**官网更喜欢称之为**GNU/Linux**，因为**GNU**提供了**bash**和**GCC**等操作系统所必须的东西，**Linux**操作系统包涵了**Linux**内核与其他自由软件项目中的**GNU**组件和软件，甚至“**linux**操作系统”之中**GNU**计划的代码所占的比重要远远高于**Linux**内核，现在所谓的各种**Linux**发行版其实应该叫做一个带有**Linux**的**GNU**系统。^[^8]现在越来越多的**Linux**发行版也认为自己是“**GNU/Linux**”。
- **Linux**发行版： 有两种说法，一种是使用了**Linux**内核就叫**Linux**发行版，一种是只有**GNU/linux**才可以叫**Linux**发行版^[^9]，**Linux**发行版之间的区别就是预装软件、软件包管理、驱动、内核补丁、桌面环境的不同。
 - **Linux**发行版有社区版和商业版之分，前者是社区维护，后者嘛，就是公司维护咯。
 - 之所以会有发行版的区别，主要还是侧重点不一样，目标用户不一样，风格不一样，背后的哲学思考不一样，
 - **Ubuntu**注重的是功能齐全，桌面华丽，预装驱动全，基本上安装了就能用口。
 - **debain**比较注重开源精神，系统里面就很少有预装闭源的软件或者驱动为了追求稳定性，软件包也不是很新
 - **Redhat**就是注重服务器系统的稳定性了，各种对kernel的补丁保证了稳定性，付费使用。
 - **centos**就是Redhat的每半年复制版（剔除了些许Redhat专有代码），继承了Redhat稳定性的特点，又由社区维护，免费用（但是2021年之后**centos8**就停止维护了，就挺震惊的口♀口）。
 - **Fedora**由红帽赞助，红帽各种新功能会先扔到Fedora上测试，算是红帽的先期测试版。
 - **Deepin**就比较牛逼了，基于debain，桌面环境用起来比较舒服，为国人的使用习惯做了优化，预装的软件很良心，自带软件商店且软件商店丰富度尚可，自带图形化包安装器，最新版甚至可以直接安装安卓软件，极大限度的照顾了小白用户，但就笔者使用下来的使用体验来看，照windows这种专业的图形界面操作系统还有相当长的差距。
 - **arch**系统就是两个字简洁，三个字，个性化，当然了，滚动更新还更新的特快口，稳定性就差点儿事儿了口。^[^10]
 - 还有一些很轻量化的系统，比如安装在树莓派上的树莓派专属**debain**就可以装在一个8g的内存卡上使用口♂口，关键是还带桌面。**TinyCore Linux**^[^11]就不到20MB，竟然还有图形化桌面环境，且**kernel**放在内存中运行，惊人不口？

- **Android**, 这个比较有争议, 争议的核心是Linux发行版的定义问题[^9], Android使用了Linux的内核, 但是Android并没有使用GNU套件, 至少Android绝对不是GNU/Linux。[^12]

- 不同的包管理系统：采用不同的包管理系统，各发行版的维护者对系统的追求不一样，比如准求稳定的发行版的仓库的软件通常是稳定版，追求新技术的发行版的仓库的软件通常是比较新的版本，不同的包管理系统有不同的使用方法和社区用户。
 - **什么是包管理系统**：包管理系统的作用是方便安装、升级和自行解决依赖，「仓库」有助于确保代码已经在你使用的系统上进行了审核，并由软件开发者或包维护者进行管理。[^13]
 - **包管理系统的区别**：有的包管理系统是图形化界面（GUI），有的包管理系统是命令行界面（CLI），命令行界面虽然使用方式不同，但笔者在使用体验上没有感觉到明显的不同口々，apt和yum都能很好的解决依赖问题，也都能方便的去安装、升级和卸载包。据说不同的包管理系统在处理依赖的能力上是有区别的^14。
 - **为什么包管理系统没有统一**：主要是历史原因。大约在同一时间建立了多个软件包管理系统，特别是.rpm和.deb。每个都有自己的拥护者，每个都足够优秀口^15，以至于没有任何一个软件包管理者具有明显的优势。发行商肯定不会完全重建他们的系统以实施其他程序包管理器。比如笔者我就比较喜欢yum，因为我单纯觉得yum的命令比较好记和简洁，apt虽然使用起来比较贴合直觉，但早些年的dpkg在安装本地的包的时候竟然不会解决依赖问题，需要用gdebi来解决口（当然，新版的debain可以用apt-get来安装本地包并且解决依赖了口）
 - **主要的包管理系统使用方式的区别**：这个随便一搜就出来了，贴个随便搜的链接[Linux 包管理基础：apt、yum、dnf 和 pkg - 知乎 \(zhihu.com\)](#)
 - **遇到仓库没有的软件怎么办**：一是自己编译源代码，比如用make命令编译，二是添加源，添加源，添加的源里面就收入了这个软件。贴一个常用的清华源（注意：这是镜像源，理论上来说，原版源没有的软件，清华的镜像源里面也没有）[centos | 镜像站使用帮助 | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror](#)

- **Android**:

- 名词解释（绝大多数人撕逼的地方就在这里）：安卓有很多含义
 - **AOSP**：Android Open Source Project Android 开放源代码项目，遵循阿帕奇协议，谷歌是该项目的维护者。
 - **商标**：谷歌公司注册了，属于谷歌公司。[^16]
 - **Android**：这个安卓指的是谷歌公司发布的包含GMS服务和各种谷歌软件和新特性的操作系统，Android=AOSP+GMS+新特性。
- 开源还是闭源：
 - Android是运行于[Linux kernel](#)之上，但并不是[GNU/Linux](#)。
 - Android默认情况下并没有本机[X窗口系统](#)，也不支持整套标准[GNU](#)库。Android为了达到商业应用，必须移除被[GNU GPL](#)授权证所约束的部分。
 - 只有基础的**Android**操作系统（包括一些应用程序）才是开源软件，任何厂商都不须经过Google和开放手持设备联盟的授权随意使用Android操作系统；大多数Android设备都附带着大量的专有软件，例如是[Google 移动服务](#)，当中包括[Google Play](#)商店、Google搜索，以及[Google Play](#)服务 — 那是一个提供与Google提供的服务[应用程序接口](#)集成的软件层。这些应用程序必须由设备制造商从Google得到许可
 - Android的硬件抽象层是能以封闭源码形式提供硬件驱动模块。HAL的目的是为了把Android framework与Linux kernel隔开，让Android不至过度依赖Linux kernel，以达成“内核独立”（kernel independent）的概念。这样安卓剥离了Linux内核以外的东西，只需要对Android修改过的Linux内核采用[GPL](#)协议，硬件驱动可以闭源[^17]，Android其余的可以采用阿帕奇协议，保护了硬件厂商的权益和安卓**GMS**和安卓其他新特性，这也就是为什么华为可以使用**ASOP**却不能使用**GMS**和**Android12**。
- **shell**: 壳程序，通俗的讲是命令解释器[^18]，他的作用就是将用户的指令告诉操作系统的核心（**kernel**），然后呢**kernel**调用电脑的各种软硬件来达到你的指令的要求。Windows的cmd是shell，Bash是Shell，powershell是shell，甚至windows的文件资源管理器也是shell（图形界面shell）^19，sh也是shell（地球上第一个比较流行的shell简称sh，是一个外国人开发出来的，简称Bourne shell口）[^20]
 - 分为图形界面**shell**和文本**shell**，当然，这个区分是我瞎编的，如有雷同，不胜荣幸哦口
 - **bash**：地球上所有能叫出名字来的Linux发行版默认使用的shell，可以说bash是sh的宇宙无敌增强版[^21]，知乎上竟然有一个问题linux初学者是学csh还是tcsh还是sh还是zsh的，我晕，初学者当然是学**bash**啊，虽然是大同小异，可实在是没有必要让一个初学者第一个学习的**shell**竟然不是**bash**。其实吧，问这个问题的同学，你甚至可

以学习在linux上用powershell, powershell是完全有能力向kernel解释你的指令的，当然啦，在linux上使用powershell是一件可以但没有必要的事情。

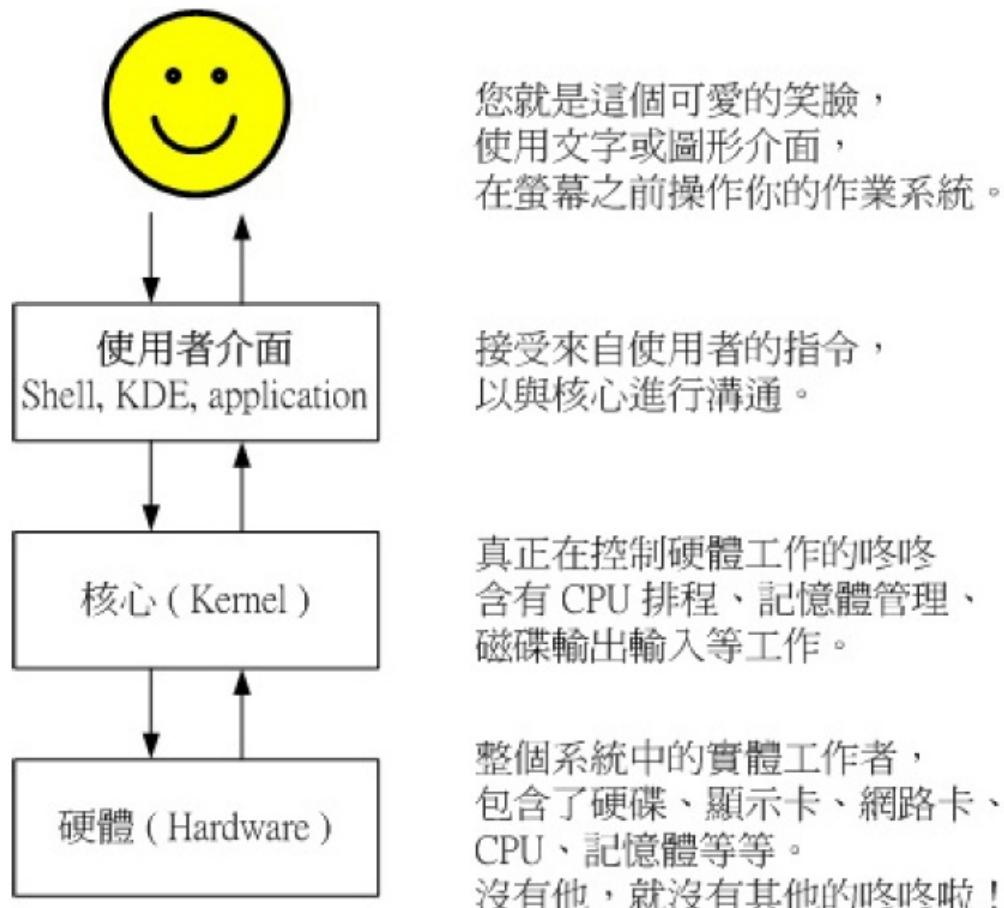
- 不同shell的区别？ 首先是功能上的区别，有的shell的自定义程度很高，可以自定义彩色显示，有的shell可以自动建议和自动补全命令，有的shell会匹配历史命令，这是功能上的区别。然后就是标准的区别，也就是是否支持**POSIX**标准，比如说fish的早期版本不支持这个标准，fish无法执行含有`&&`的命令，这对使用者造成了很大的困扰。

(一) 先从UNIX到GNU讲起

这部分很简单，20世纪六十年代，贝尔实验室，开发出了unix。Unix不是开源软件，Unix源码可以与它的拥有者**AT&T**通过协议获得许可证。第一个已知的软件许可证在1975年卖给了伊利诺伊大学。

Unix在学术界发展迅速，随着伯克利成为重要的活动中心，在70年代，Ken Thompson（开发unix的领导者）有一个学术休假。通过在伯克利的Unix的所有活动，一个新的Unix软件支付诞生了：伯克利软件发行版，或者叫**BSD**。^[^4]

有好多公司和团体吧，就在unix的基础上二次开发，整出来了好多unix的分支，并且吧，这些个分支所使用的shell还不一样，shell是什么呢？他的官方名称就是内容解释器或者叫壳程序，他的功能就是把用户输入的命令告诉操作系统的核心，然后这个核心再指挥硬件。



shell不一样，进行相同的操作接收的命令也就不一样，也就是说，相当于这些shell所能听懂的人话不一样，好比shell是不同国家的人，有的shell只能听懂英文，有的shell只能听懂日文，这可咋办，这地球上unix分支这么多，甚至不同的计算机安装的分支都不一样，总不能让用户换个电脑还得重新学门语言吧，于是乎**POSIX**标准就出来了，不管你是谁，都应该能听懂同一种语言，全世界都得说中国话（当然这只是戏虐）！于是乎，同样的命令，可以在所有支持**POSIX**标准的**shell**上运行。

然后，[理查德·斯托曼](#)，我们就暂且叫他老李头吧，老李头对unix很不爽，于是乎呢，他有了一个想法，那就是开发一个完整的开放源代码的类unix操作系统，来取代unix。他给自己的这个想法起了个名字叫**GNU**，名字的意思呢是“**GNU's Not Unix!**”，翻译成中文就是“老子不是unix，老子叫**GNU**！！！”他这个想法一提出来就得到了广大人民群众自发的支持，毕竟大家早就看unix不爽了，unix，啥都不是！想学习学习你的源代码竟然要花钱，老子不用你了！于是大家怀着这个要将unix干翻的想法开始工作起来了。

老李头在接下来的几年，成立了自由软件基金会，开始雇佣人来写代码，并且发表了操作系统界的《独立宣言》——**GNU**宣言。还整了个**GPL**协议，这个协议挺**niubility**啊，遵循这个协议的软件必须开源，并且使用遵循**GPL**协议代码的软件也必须开源，哪怕是你商用，也得开源。



这个计划吧，一开始还算顺利，向命令解释器（bash）啊，编辑器（Emacs）啊，编译器（GCC）啊啥的都进展的挺顺，可是啊，就是核心开发不出来，核心是啥呢？上面提过一嘴，核心就是能够控制硬件来执行接收到的命令的东西。这个东西可把老李头愁坏了，他是吃也吃不下，睡也睡不着，头发大把掉，胡子蹭蹭长（只是在调侃理查德的大胡子）……

（二）天降正义，Linus！

时间来到了九十年代，一个芬兰赫尔辛基大学的大学生林纳斯，我们管他叫小林吧，这个小林不声不响，开发出来了操作系统内核linux，并且这个Linux还采用了**GPL**协议，小林按当时的话来说，就是比较帅呆了，还特谦虚幽默，你看他当年放出Linux时说的那话

我在做个（自由的）操作系统（就是个兴趣爱好，我不会搞得像GNU那么大那么专业），打算让它工作在386 AT平台上。它从四月就开始酝酿了，马上就快好了。我想要那些喜欢或不喜欢minix的人的意见，因为我的系统和它有点类似（同样的文件系统的物理布局——由于实际原因——还有些其他的东西）。

我已经移植了**bash**(1.08)和**gcc**(1.40)，而且看起来奏效了。这意味着我会在几个月内得到一些实用的东西。“……”是的——它没有任何minix代码，并且它有一个多线程的fs。它不可移植（使用386任务切换等），而且它可能永远不会支持除AT硬盘之外的其他东西，因为我只有这些:-)。

“……”它基本上是用C语言写的，但是大多数人可能不会把我写的东西叫做C语言。它使用我能找到的386的每个可以想象的特性，因为它也是一个教我关于386的功能的项目。我前面提到过，它使用内存管理单元来进行分页（还没实现到对硬盘的功能）和分段。这个分段功能使得它真正的依赖于386（每个任务都有64Mb的代码和数据段——4Gb中最多64个任务。如果有人需要超过每个任务64Mb的限制，那将是个麻烦事）。“……”我的一些C语言文件（特别是mm.c）几乎用了和C一样多的汇编。“……”不像minix，我也碰巧喜欢中断，所以中断将在不试图隐藏背后的原因的情形下被处理。[^22]



林纳斯·托瓦兹，Linux内核首创者。

老李头看这个小林是越看越喜欢，小林呢觉得GNU也不错，于是呢，GNU+linux kernel（内核）=GNU/Linux！

（三）越来越受欢迎

从GNU/Linux第一个发行版Boot-root问世以来，GNU/Linux越来越受到欢迎，Marc Ewing成立Red Hat软件公司，成为最著名的Linux经销商之一，也证明了GNU/Linux在商业上被认可（GNU/Linux虽然开源，但不代表不可以商用，Red Hat是开源的，但是并不免费，Red Hat会为Red Hat的用户有偿提供技术服务）。

现在，整个GNU/Linux操作系统家族基于Linux内核部署在传统计算机平台（如个人计算机和服务器，以Linux发行版的形式）和各种嵌入式平台，如[路由器](#)、[无线接入点](#)、[专用小交换机](#)、[机顶盒](#)、[FTA接收器](#)、[智能电视](#)、[数字视频录像机](#)、[网络附加存储（NAS）](#)等。工作于[平板电脑](#)、[智能手机](#)及[智能手表](#)的Android操作系统同样通过Linux内核提供的服务完成自身功能。尽管于[桌面电脑](#)的占用率较低，基于Linux的操作系统统治了几乎从移动设备到主机的其他全部领域。截至2017年11月，世界前500台最强的[超级计算机](#)全部使用Linux。

GNU/Linux被打包成供个人计算机和服务器使用的Linux发行版，一些流行的主流Linux发行版，包括[Debian](#)（及其派生版本[Ubuntu](#)、[Linux Mint](#)）、[Fedora](#)（及其相关版本[Red Hat Enterprise Linux](#)、[CentOS](#)）和[openSUSE](#)等。Linux发行版包含Linux内核和支撑内核的实用[程序](#)和库，通常还带有大量可以满足各类需求的应用程序。^[^23]

（四）命名之争，linux还是GNU/linux？

有越来越多的人管GNU/linux叫linux操作系统，老李头团队的某些人一看不对啊，这怎么行？虽然使用了Linux内核，可也不能抹杀GNU的贡献吧？GNU起初可是要打造GNU系统的啊。

支持GNU的人认为GNU提供了**bash**和**GCC**等操作系统所必须的东西，Linux操作系统包涵了[Linux内核](#)与其他自由软件项目中的GNU组件和软件，甚至“[linux操作系统](#)”之中**GNU计划**的代码所占的比重远高于[Linux内核](#)，现在所谓的各种Linux发行版其实应该叫做一个带有[Linux](#)的**GNU系统**。^[^8]

还有些人呢，就发话了，一个操作系统最重要的是啥？不就是内核吗？我们就叫Linux怎么了？（不代表笔者看法口口口）

“GNU/Linux”此名称是**GNU计划**的支持者与开发者，特别是其创立者[理查德·斯托曼](#)对于Linux操作系统的主张。由于此类操作系统使用了众多GNU程序，包含[Bash（Shell程序）](#)、[库](#)、[编译器](#)等等作为Linux内核上的系统包，理查德·斯托曼认为应该将该操作系统称为“**GNU/Linux**”或“**GNU+Linux**”较为恰当，但现今多数人仍称其为Linux。就1997年之前的Linux来看，一间CD-ROM的供应商所提供的资料显示在他们的“Linux发行版”中，GNU软件所占最大的比重，大约占全部源代码的28%，且还包括一些关键的部件，如果没有这些部件，系统就无法工作，而Linux本身占大约3%。

Linux社群中的一些成员，如[埃里克·雷蒙](#)、[林纳斯·托瓦兹](#)等人，偏好Linux的名称，认为Linux朗朗上口，短而好记，拒绝使用“**GNU/Linux**”作为操作系统名称。并且认为Linux并不属于**GNU计划**的一部分，斯托曼直到1990年代中期Linux开始流行后才要求更名。有部分[Linux发行版](#)，如[Debian](#)，采用了“**GNU/Linux**”的称呼。但大多数商业Linux发行版依然将操作系统称为Linux。而有些人则认为“操作系统”一词指的只是系统的内核(Kernel)，其他程序都只能算是[应用软件](#)，因而，该操作系统应叫Linux，但Linux系统包是在Linux内核的基础上加入各种GNU软件包集合而成的。^[^24]

反正吧，这事儿现在也没个准确的说法，我觉得吧，人GNU是做了很大贡献的，这个肯定都承认，可是挡不住Linux这个叫法朗朗上口写起来还省墨水啊，总之，现在（2021年8月31日）各种官方还没有个统一的说法，叫啥名字都无伤大雅，不管什么名字，都掩盖不了GNU和linux kernel的成功，也掩盖不了Linux操作系统对这个世界（尤其是开源世界）的贡献。

以上，就是Linux从无到有的历史了，通过这些历史，我们将不再对unix, GNU, Linux, Shell和bash等概念感到一筹莫展了，这就足够了。

现在，可以回到文章开头，去阅读我建议你跳过的名词解释+吐槽部分了。

``(``▽``)Bye~Bye~

参考资料

[^2]:[Linux与Unix区别在哪里 | 《Linux就该这么学》 \(linuxprobe.com\)](#)

[^3]:[posix是什么都不知道，还好意思说你懂Linux？ - 知乎 \(zhihu.com\)](#)

[^4]:[Linux与Unix区别在哪里 | 《Linux就该这么学》 \(linuxprobe.com\)](#)

[^5]:[GNU计划 - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

[^6]:[Linux内核 - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

[^7]:[GNU计划 - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

[^8]:[Linux 和 GNU - GNU 工程 - 自由软件基金会](#)

[^9]: 可以把Android说成是Linux发行版吗？ - SegFault的回答 - 知乎

<https://www.zhihu.com/question/383947288/answer/1361731747>

[^10]:[Linux 各大发行版有什么特色？ - 知乎 \(zhihu.com\)](#)

[^11]:[Tiny Core | linux发行版 \(linux265.com\)](#)

[^12]: 可以把Android说成是Linux发行版吗？ - AidenLeong的回答 - 知乎

<https://www.zhihu.com/question/383947288/answer/1119467476>

[^13]:[Linux软件包管理基本操作入门 | 《Linux就该这么学》 \(linuxprobe.com\)](#)

[^16]: AOSP (Android Open-Source Project) 跟 Android 是何关系？ - yaui的回答 - 知乎

<https://www.zhihu.com/question/21448269/answer/1888531716>

[^17]:[Android - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

[^18]:[鸟哥私房菜 - 第十章、认识与学习BASH \(vbird.org\)](#)

[^20]:[鸟哥私房菜 - 第十章、认识与学习BASH \(vbird.org\)](#)

[^21]:[Linux SHELL中sh和bash的区别 - 御用闲人 - 博客园 \(cnblogs.com\)](#)

[^22]:[Linux内核 - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

[^23]:[Linux - 维基百科，自由的百科全书 \(wikipedia.org\)](#)

扫盲教程，本人能力有限，如有错误，请于评论区指出，不胜感谢，勿喷。

[TOC]

扫盲

什么是vscode?

Visual Studio Code是一个轻量级但功能强大的源代码编辑器，可在您的桌面上运行，并且可用于Windows, macOS和Linux。它具有对JavaScript, TypeScript和Node.js的内置支持，并具有丰富的其他语言（例如C, C#, Java, Python, PHP, Go）和运行时（例如.NET和Unity）扩展的生态系统。。通过这些入门视频，从VS Code开始您的旅程。

——节选自vscode官网

什么是git

Git 和其它版本控制系统（包括 Subversion 和近似工具）的主要差别在于 Git 对待数据的方式。从概念上来说，其它大部分系统以文件变更列表的方式存储信息，这类系统（CVS、Subversion、Perforce、Bazaar 等等）将它们存储的信息看作是一组基本文件和每个文件随时间逐步累积的差异（它们通常称作 基于差异（**delta-based**）的版本控制）。

——节选自Git官网

也就是说vscode是一个编辑器，即使功能强大，也依然是编辑器。Git是版本控制软件，这个软件的特色是他储存版本的方式是去储存版本间的差异

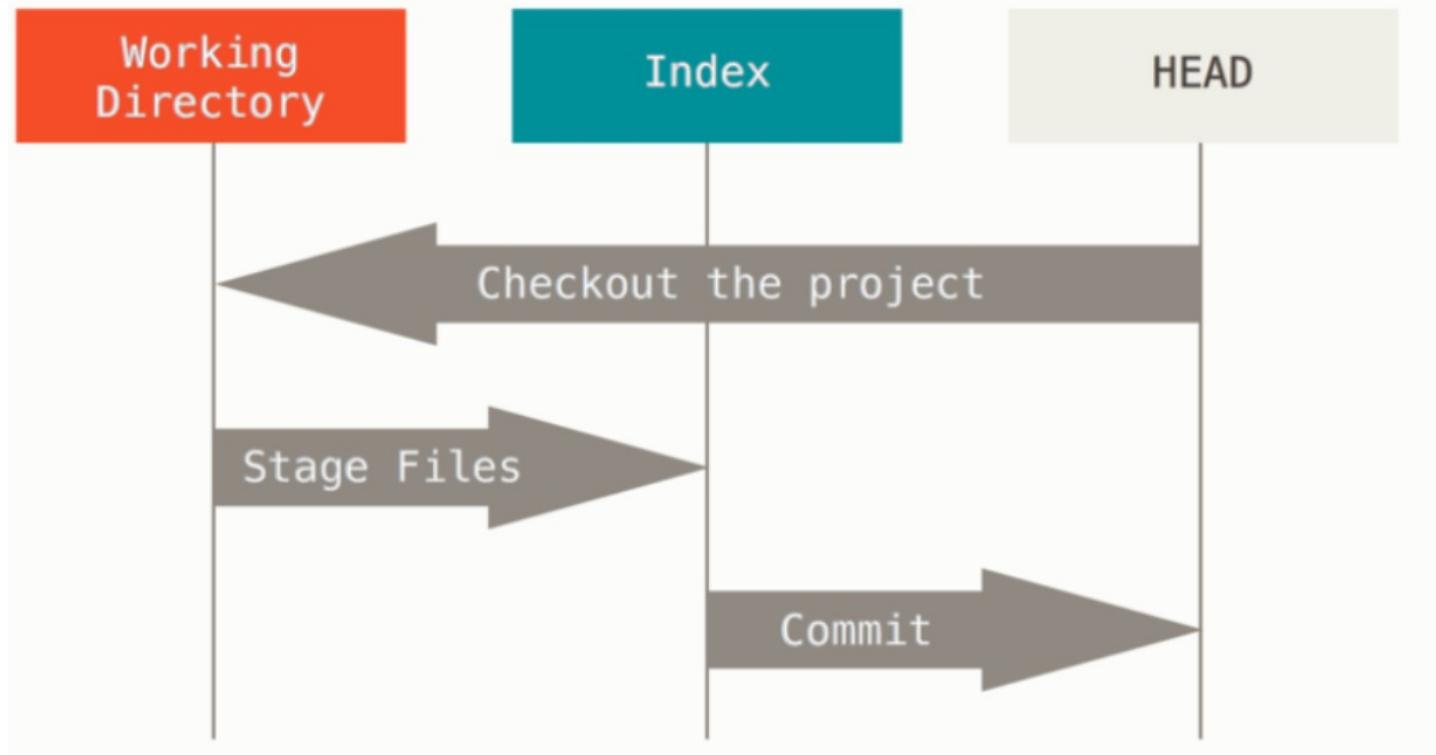
Git概念简单解释

既然要玩git了，建议了解git的命令行，[Git - 获取 Git 仓库 \(git-scm.com\)](#)，这里不对命令行做过多解释。只说一下git的某些概念
git的第一个概念是仓库，仓库里面有很多文件，仓库也记录了所有文件的版本(被提交过的)。

一个文件的生命周期有三个阶段，分别是 `work dir`、`HEAD`、`Index`。三者关系是进阶的，一个文件先由 `work dir` 上传到 `index`，再由 `index` 上传到 `HEAD`，再上传到 `HEAD` 之后，就会自动生成一个版本号，vscode里面将文件从 `work dir` 上传到 `Index` 的过程称之为暂存，将 `Index` 上传到 `HEAD` 的过程称之为提交

工作流程

经典的 Git 工作流程是通过操纵这三个区域来以更加连续的状态记录项目快照的。



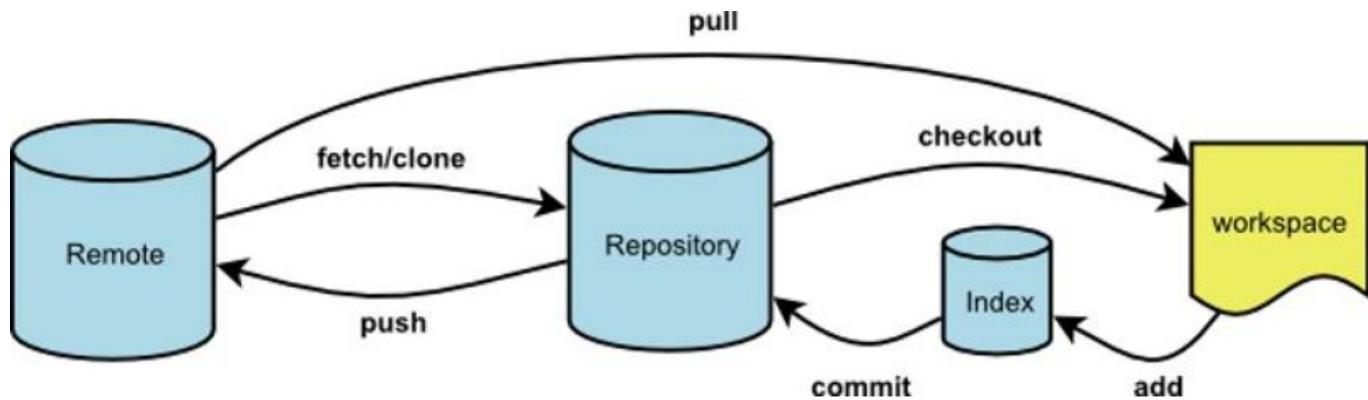
版本回退

Git里面有`work dir`、`Index`和`Head`的很大一个好处是可以方便的进行版本回退和撤销提交的操作。[git reset 和 checkout 以及 HEAD COMMIT ADD详解 - sogeisetsu - 博客园 \(cnblogs.com\)](#)

	HEAD	Index	Workdir	WD Safe?
Commit Level				
reset --soft [commit]	REF	NO	NO	YES
reset [commit]	REF	YES	NO	YES
reset --hard [commit]	REF	YES	YES	NO
checkout <commit>	HEAD	YES	YES	YES
File Level				
reset [commit] <paths>	NO	YES	NO	YES
checkout [commit] <paths>	NO	YES	YES	NO

远程仓库

比较著名的远程仓库有GitHub, gitee等。远程仓库保证了可以将本地文件储存到云端，保护了文件，并且方便协作。



vscode对git的操作

首先需要下载两个软件一个叫vscode、另一个叫GIT（废话）。下载链接如下口

- [Visual Studio Code - Code Editing. Redefined](#)
- [Git - Downloads \(git-scm.com\)](#)

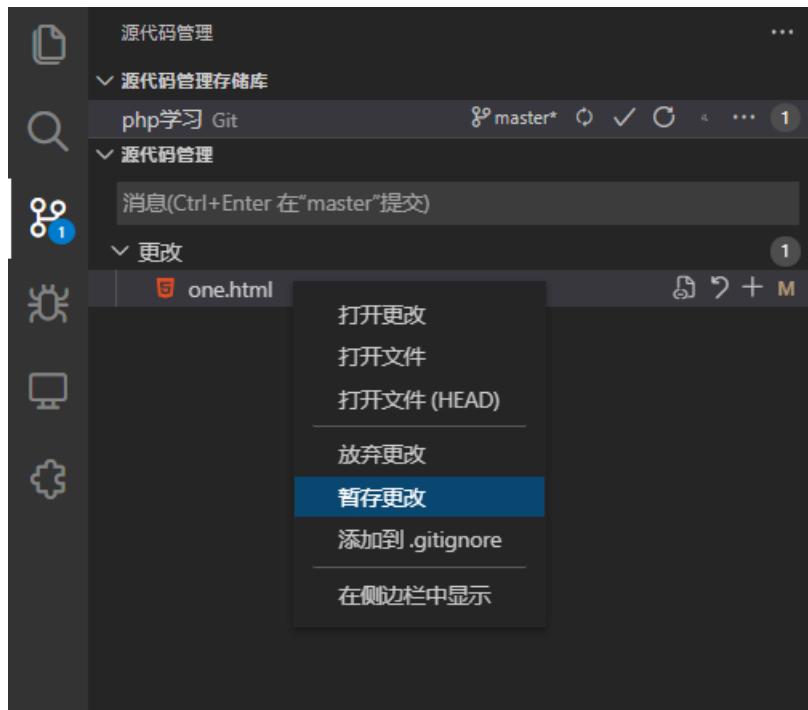
初始化一个仓库

用vscode打开一个文件夹，在vscode的命令面板 (`CTRL+shift+p`) 输入`git init`将会出现选项点击即可。

php学习 - Visual Studio Code

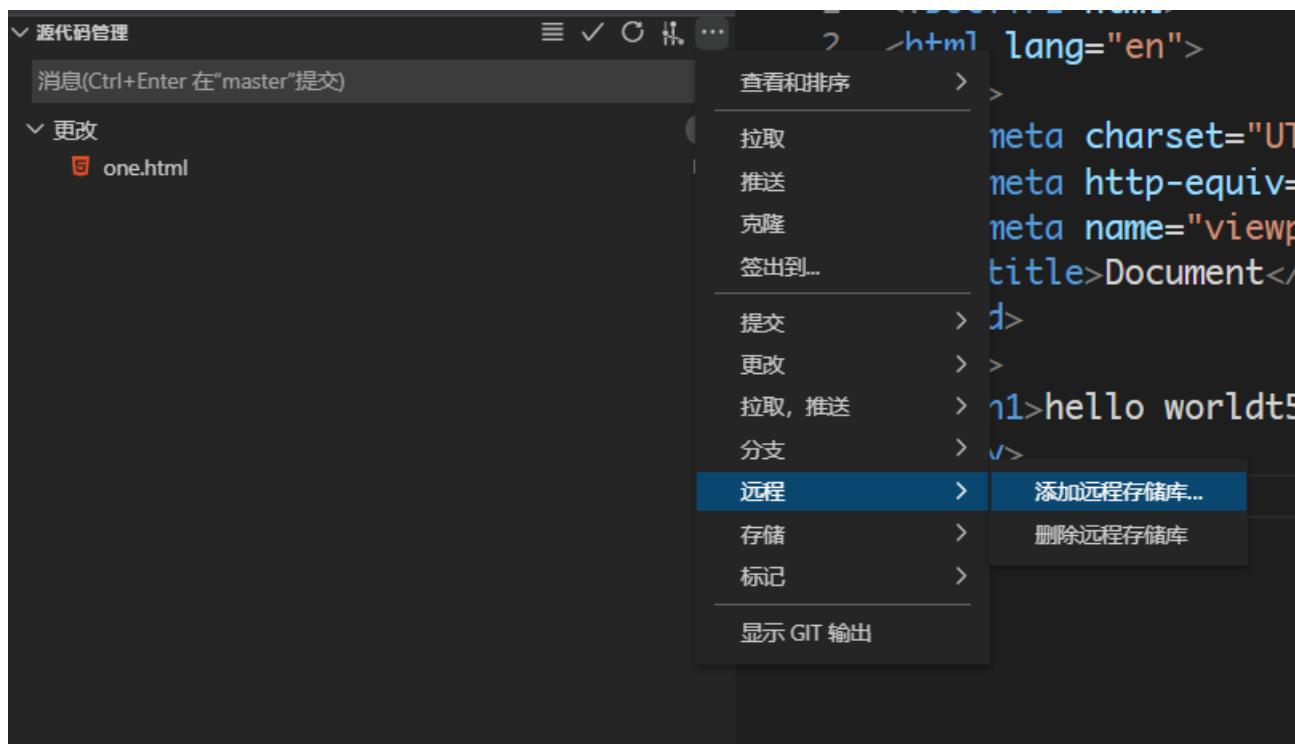
暂存和提交

左边dock栏，源代码管理，选定文件，右键进行暂存和提交的操作，这个比较简单，按文字提示来就是了。



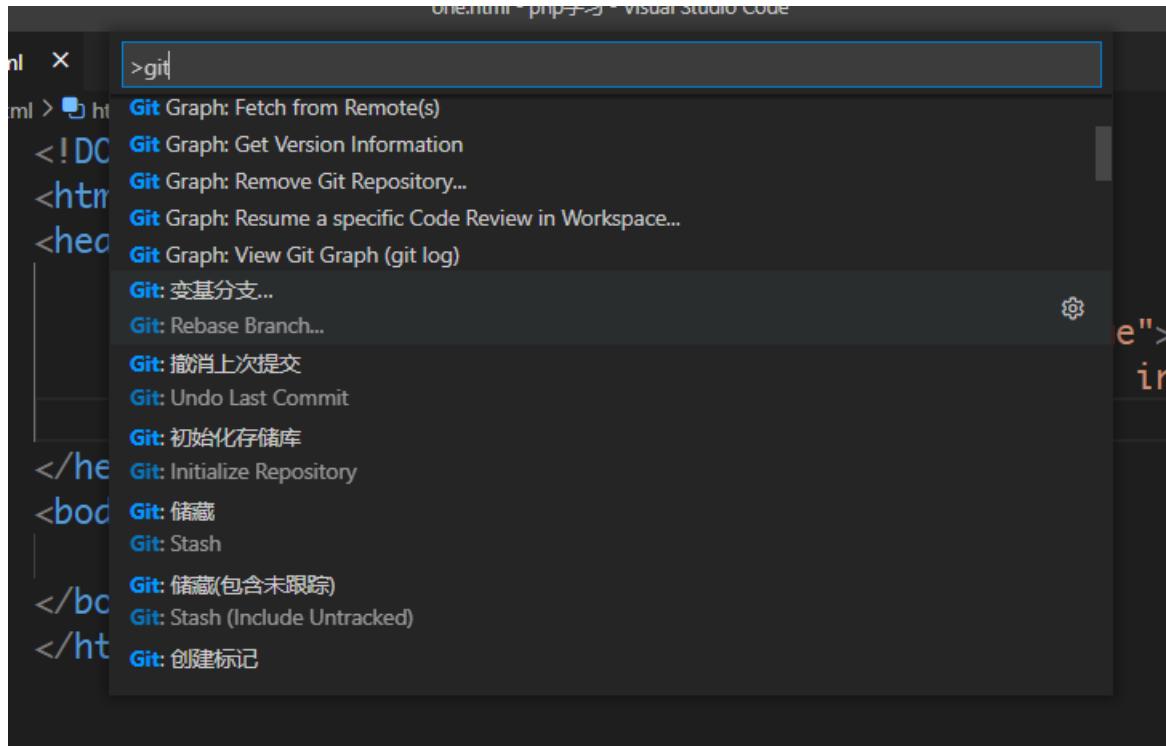
链接远程仓库

按下图操作，随后填git链接或者选择直接从GitHub中创建，全程傻瓜化操作，狂点下一步就完事儿了。

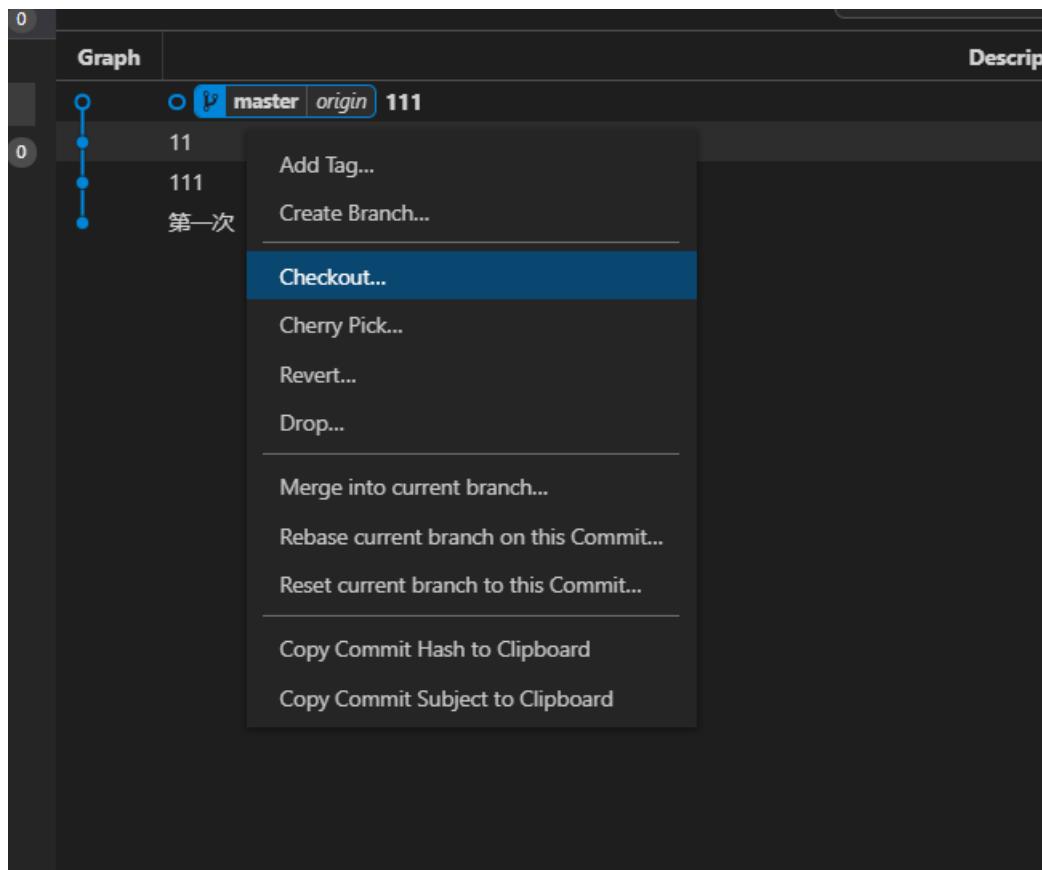


版本回退

两个方法，一个是在命令面板进行撤销操作。



第二个方法是安装插件，[Git Graph - Visual Studio Marketplace](#)，右键点击checkout进行版本回退



——
大家好，我是墙墙，我，是，青岛科技大学表白墙。

这个视频，应广大同学地要求，我们要表达我们的愤怒，戳穿学校的谎言，在这之前，我们通过各种方式来维权，我们的表白墙被限流，我们的微博话题主持人被收买，我们的同学被出卖，被抛弃，被学校用极其不体面的方式约谈。

万般无奈，我们只能用这种方式来告诉广大的同学们，青岛科技大学发生了什么。我们只能用这种方式告诉学校的领导，学生在发声，我们，还活着。我们只能用这种方式来告诉千万大学生们，有人在勇承重担，有人，在奔走呼号。

——
二

墙墙最大的职责就是帮助学生们发声，在2021年7月9日开始，墙墙收到了密集的投稿，这些投稿都指向了同一个事件，。。。青岛科技大学宿舍八改六事件，经过梳理，我发现了一个骗局，一个本不应该存在于这个世纪的骗局，我发现了一场虚假的表演，在这场表演当中，学生们被无情地代表，学生们的权益被践踏，被侮辱。

现在，我尽量用客观公正的语言，去阐述这个事件的起末，在阐述的过程中，我会引用一些同学的投稿，迫于斗争形式的需要，我无法将你们的名字公布出来，我只能，在这里向无声的英雄致意。

青岛科技大学，是一所坐落于青岛，号称“三地五校”的综合性大学。青岛科技大学四方校区，是青岛科技大学历史最悠久的校区，四方校区因建校时间过长，学生宿舍条件一直较差。



有趣的是，学校在对外宣传的过程中，学校一直宣传的是崂山校区的宿舍，崂山校区，六人宿舍，有阳台，有桌子。直到今天，学校招生官网上也是这么宣传的，对四方校区宿舍的情况却言辞闪烁，丝毫不提学校宿舍床铺老旧，墙面漆黑，走廊闷热潮湿，一股厕所的味道，宿舍虫蚁不断，因为没有阳台，女生们只能将内衣内裤挂在窗户外面，就像19世纪的美国贫民窟一样，毫无

隐私可言。学校一边宣传着四方校区传统的强势的专业，一方面对化工学院的居住环境闭口不提。一方面说着学校经费紧张，一方面声称要在淄博出资16亿建设淄博校区。这些，学生们都忍了，学生们都对学校所面临的情况表示理解。即使是在炎热的夏天经常有人中暑，有些女生因为太热身上起了汗斑，这些，我们都忍了。

事情是这样的。

26号公寓楼建成，由于旧宿舍楼八人间而且环境破旧，而新楼宿舍空间大且配有空调，新老宿舍的严重差异形成了宿舍分配问题，住在老楼的学生向辅导员询问是否能够住进26号楼，并且表明了住进26号楼的强烈意愿。

26号公寓楼建设时长达到或超过1年，封堵了慧园一侧的门以及部分道路。我校女生在慧园居住在此门被封堵后，去体育场，教育超市，实验室，第一教学楼，博苑食堂等地需要大量绕路。26号楼建设过程中，产生大量噪音，在中午，晚间，半夜，凌晨等时段经常性进行施工，而声音不能被有效隔除，21号楼，22号楼，22号甲，慧园甲、乙等公寓居住的学生休息经常收到严重性影响，而这种情况无人管理，学生休息质量极差，在心理及生理层面受到严重伤害。而将宿舍“8改6”的措施并没有使学生的居住环境得到改善，宿舍依旧不具有有效的降温措施及优良的住宿设备配备。对学生的正常生活产生了很多影响。26号楼有14~15层，看足够空间选择性将旧楼的学生搬迁至其内，而学校并未有意安排大部分旧楼学生入住。学校有明显意图将宿舍楼留给新生以此来达到增大学校招生的吸引力，存在严重的不公平现象。

住在老楼学生的相关专业辅导员向上级申请，无果，上级表示将原有**8人间**改造为**6人间**，需要两位同学搬走，而宿舍的居住环境几乎没变，反而要面对哪两个同学搬走的问题。

学校并没有发布“8改6”的相关文件，而是辅导员将决策以“口头通知”的形式转述给各班班长及支书，再下发至相关同学，甚至并未有在学生群内部的书面转述。而口头通知不具效力，而学生不得不服从，以此来造成“学校并未强迫，遵从学生意愿”的假象，导致学生向外界维权，曝光时并不能拿出有效证明，学校可以以“故意抹黑学校”为理由，对向外界求助的学生给予处分或谈话。

2021年7月7号星期三(正值考试周)，学校没有任何前奏，的通知学生八改六，通知方式是通过导员给班长，开会口头传达的，不让班长发通知，只让口头给班里人传达，一直到今天(7月12日)，学校也没有发任何书面的通知或者推文，导员开会时怎么改说的很详细，唯独没有说时间，八改六的具体方案是：拆掉靠门的两个下铺，不放柜子桌子，让用这个空间放行李箱，宿舍不一定是原来的宿舍，但一定实在原来的宿舍楼，但尽量让一个班的宿舍挨着，至于要搬出的两个人让宿舍自己协商，收费按六人间标准收费，这期间有很多学生和家长提意见，但学校一直采取打压的行动，约谈学生，给学生施压，家长群里面家长提意见会被导员偷偷提出群，最后直接解散了家长群。

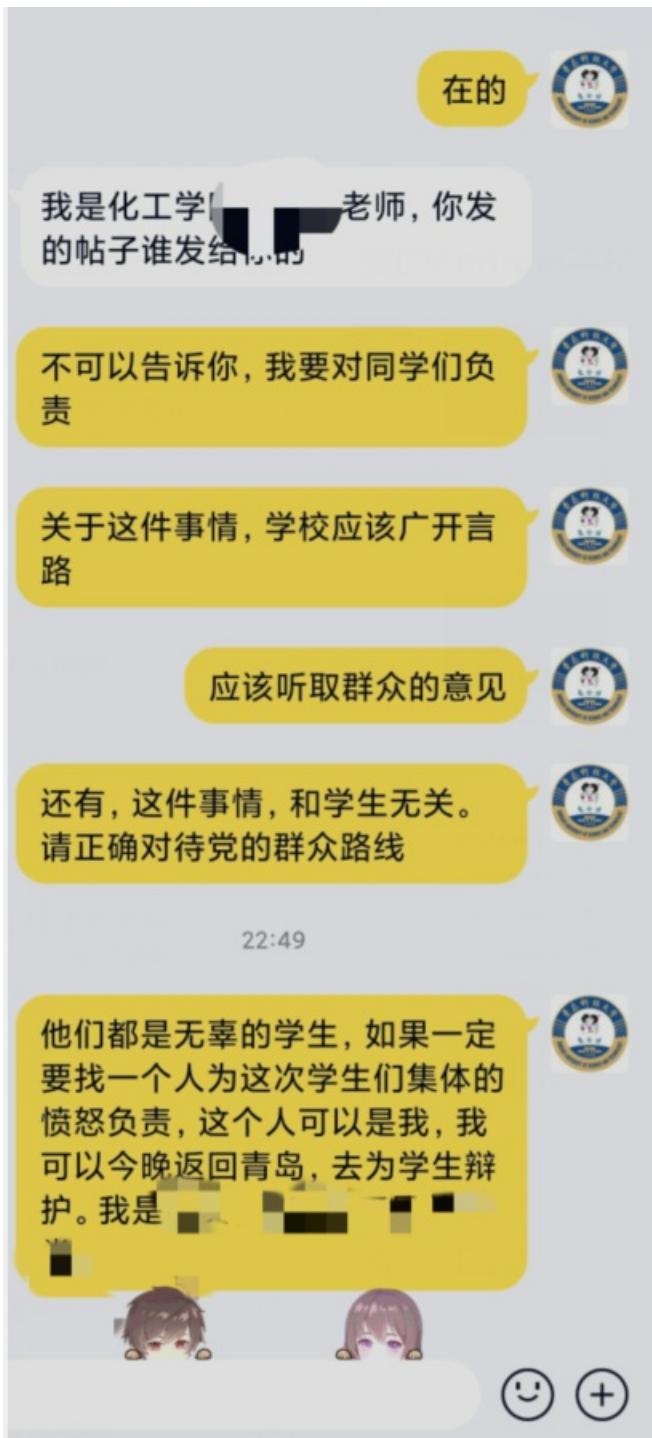
事情起因是这样的。

7月7号星期三（正值考试周），学校没有任何前奏的通知学生八改六，通知方式是通过导员给班长开会口头传达的，不让不让班长发通知，只让口头给班里人传达。一直到今天学校也没有发任何书面的通知或批文。导员开会时怎么改说的很详细，唯独没有说时间。八改六的具体方案是：拆掉靠门的两个下铺，不放柜子桌子，让用这个空间放行李箱，宿舍不一定是原来的宿舍了，但一定是在原宿舍楼，但尽量让一个班的宿舍挨着，至于要搬出去的那两个人让宿舍自己商讨，收费按六人间标准收费。这期间有很多学生和家长提意见，但学校一直是采取打压的行为，约谈学生，给学生施压。家长群里面家长提意见会被导员偷偷踢出群，最后直接解散了家长群。

可能我描述的会有遗漏，也可能会有我的主观因素在里面，恳请补充指正。

起初，学生的诉求并不是非要住进新盖的26号楼，而且学校这种不把18-19级当人看的态度，彻底让学生们寒了心，口口声声说以学生为本，橡胶精神，结果呢？每到夏天就热的睡不着了，有同学从六月中旬开始背上就起了湿疹，八个人靠着一个五个挡位却只能开到二档的风扇来度过夏天，最近青岛平均气温在30度，湿度还很高，每天都接近两点才睡着，这种日子不知道还要过多久。如果学校能认真对待22-23两栋旧楼，哪怕是空调自费，学生们都接受，学校总是用线路老化来敷衍学生，可是，传达室和宿舍同在一个楼，为什么传达室就可以有空调？难道传达室和宿舍用的不是一个线路？为什么学生这么倒霉？

学生们的投稿像雪花一样飘像了表白墙，墙将学生们的投稿悉数发表在了表白墙上，一篇家长谎称学生对学校的做法表示理解的投稿终于引发了群体的愤怒，学校的某位领导因为自己的如意算盘被打破，学校的一位领导找到了我，让我将所谓“闹事”的学生告诉他，我据理力争，没有告诉他。



紧接着，表白墙被莫名举报，有人因为评论了我们表白墙帖子而被学校约谈。

约谈，真的是一个说起来好听，实则是学校对公权力无耻滥用的一个行为，学校面对弱势的学生，说服不了学生就收买学生，收买不了学生就处分威胁学生，学校用尽了各种方法来打压学生，唯独不愿意听取学生们的意见，这样的学校，是怎样的学校？这样的约谈，是怎样的约谈？我们的学生，是怎样的不幸者与悲哀者？

一方面，学生们群情激愤，甚至有了寻短见的想法

4

来来来，约谈我，谈完我就跳楼，除了会压榨学生还会干啥，倒八辈子霉考到青科大，死了拉倒。

昨天14:36 · 回复

4

明天如果没跳是不是就是学校怂了？

昨天14:37 · 回复

4

回复 [Simon、](#)：我今晚就跳，死也要拉死青科大。

昨天14:38 · 回复

4

敬你是条汉子，7天后给你上香。

昨天14:39 · 回复

4

冷静冷静，为这学校死了不值得

昨天14:43 · 回复

4

冷静，不值得这样，还有很多爱你以及你爱的人等着你呢，我们一起发声投诉就行了，千万不要这样啊

昨天16:49 · 回复

一方面，有的学生，不知道是站在什么立场，什么角度去污名化学生们维权的行为，说什么“如果你觉得你的大学不好，你就去建设她；如果你觉得宿舍环境不好，你就好好学习将来管理学校改变她；她有缺点，我们一起修正，而不是一味的谩骂，抱怨，逃离。你所站里的地方，正是你的大学，你怎么样，青科便怎么样，你是什么，青科便是什么，你若光明，青科便不黑暗”

学生们向学校表达自己的看法，遭到了学校的所谓的善意的约谈，学校听不到学生的呼声，有些人将学生们不甘于学校的不公平说成是“谩骂，抱怨和逃离”，墙墙想说的是，如果觉得学校不好就要去建设学校，那么维权就是我们建设学校最好的方式。

如果某些既得利益者非要套用诸如“觉得学校不好，就应该好好学习成为学校的建设者”之类的话来侮辱学生们的愤怒。那么我们可以说“起初，学校只是借防疫之名来封校，我沉默了，因为我不怎么出学校；然后学校崂山校区的食堂因为外卖送不进来而借机涨价，我沉默了，因为我的家庭情况能承担得起涨价；紧接着，有人因为宿舍过于炎热而中暑，我还是沉默了，因为我的身体尚可；最后，学校要借八改六之名来强制要求我离开宿舍，这时，已经无人替我讲话了，我只能离开原有的宿舍，去一个陌生的宿舍，这个宿舍的条件仅仅是少了两张床，甚至没有多加一张桌子，我只能承担多出来的宿舍费”

如果你觉得你的大学不好，你就去建设她；如果你觉得宿舍环境不好，你就好好学习将来管理学校改变她；她有缺点，我们一起修正，而不是一昧的谩骂，抱怨，逃离。

你所站里的地方，正是你的大学，你怎么样，青科便怎么样，你是什么，青科便是什么，你若光明，青科便不黑暗！

码了

尽管有很多学生因为评论和转发我的帖子而被学校约谈，但是学生们依然向我表达了支持，这个时候，我迟疑了

在这次事件中，无论是什么原因，我都被推上了风口浪尖，我很骄傲，因为我将同学的意见表达了出来，我并不感到惧怕，风口浪尖的下面是人民群众的汪洋大海，在这样一个位置，有无数学生给我撑腰，我是安全的。可是学生们呢？我已经了解到有人因为评论了我发的帖子而被约谈，我既然做了这件事我就不怕，可学生怎么办？学生没有像我一样在公开的平台上有诸多支持，说白了，学生没有这么多的关注者，学生的遭遇无法让更多的人知道，我安全了，学生却被约谈了，这不公平。

如果学生需要发声，即使是粉骨碎身我也不怕，如果学生的前途因为我受到损害，我愿意用我的胸口来换堵枪眼。学生或许不需要我来保护，但是学生既然选择了我来发声，我就有义务保护我的关注者和评论者。

于是，我决定不再通过qq空间帮学生们维权，我决定通过匿名性更好的微博来继续帮学生们发声，我相信，学生们的声音不会消失，学校没有资格让学生们闭嘴，学生对自由和公平的追求是杀不尽的，因为真理永远存在。

我本以为，转战微博会让学生们的声音走向更大的平台，会有更多的人了解到我们的声音，我们或许会凭借热搜的压力让学校向学生道歉，没想到，我们走进了一个更大的漩涡，一个更加吃人不吐骨头的修罗场，我们真正地见识到了社会的险恶。

微博话题“青岛科技大学八改六人间”的话题主持人校园咨询君总是没有任何理由的删除学生们的投稿，用删微博的方式删掉涉及八改六问题本质的帖子，学生们的声音，无法通过微博表达出来。后来有学生投稿说，校园咨询君通过碰瓷学校收了一笔钱，现在到了压热度的时候了。

7:49

4G

< 返回

...

转发 6 评论 11

赞 43



你为什么要删青科大表白墙的八改六?

7-12 19:15

回复 赞 476

这个校园咨询君也是个内鬼

这家伙这个资讯君真是营销号，给钱就删

校园咨询君就是专门碰瓷学校，找学校拿钱的

直接改名叫四方公关学院吧

领导权利大没办法

三

现在，QQ被限流，微博话题主持人被收买，我们的维权之路眼看就要失败。

学校因为我们之前的维权，终于出了一张没有公章的所谓的公告

四方校区“8改6”学生宿舍改造实施计划

为做好宿舍“8改6”工作，改善学生住宿条件，后勤处将利用暑期对宿舍实施以下改造项目：

一、拆除房间门口的两张下铺床板和脸盆架，腾出空间，方便同学放置大件行李和物品，扩大学生室内活动空间；

二、为两个贴地橱柜加装隔板，学生可作为公共鞋柜或者其他公用储物柜。

三、对宿舍家具进行维修维护，加固书架，对宿舍内大桌子进行翻新维修更换。

四、对房间内开关、插座、照明灯进行维修更换，插座更改为5孔，照明灯更换为LED灯管，改善房间照明。将走廊灯更换为LED吸顶灯，改善公共区域照明。

后勤管理处

2021.7.12

这篇公告，不疼不痒，不回答学生们最关注的问题。

我们的维权之路，我们的发声之门被堵死。导员在考试周开会，意欲借助考试周这个机会强推八改六，打学生一个措手不及。校长信箱，形同虚设，对的，我说的是形同虚设。如果说学校现在是一个火坑，那么校长陈先生就是致学生于水火之中而不顾的“独夫民贼”。

现在，我们只能通过视频的形式来向社会反应情况，来表达我们的诉求，这不会是我们最后的维权，这只是一个开始。

我们胜利了吗？没有

我们失败了嘛？也没有

我们勇敢的发出我们自己的声音了吗？是的

学校妥善解决八改六问题了吗？并没有

被约谈的学生得到道歉了吗？没有

学校的谎言被戳破了吗？是的

但是，这远远不够

我们诉求如下

1. 还校于民，纵观我校历史，学生占据了学校的大多数，学校的重大决定自然应该交由学生们来做最后的决定。学生是学校历史的主体，学生应该得到重视。这才是真正的群众路线。
2. 由家长代表和学生代表和校方统一谈判八改六事件，学校必须承认八改六事件的伟大意义，是否八改六，怎么八改六？必须得到家长和学生的一致认同。
3. 如果八改六，宿舍不应该只是抽掉两个床板完事，至少应该安装桌子或者柜子
4. 暂缓八改六，现在是考试周，应该在考完试之后，下学期开始再进行八改六的工作*

大家好，我是墙墙，专注为学子发声。

这只是一个开始，不会是结束，感谢一直支持我们的同学，我向你们致意。

本视频如果火了起来，如果产生了收益，收益将会用来给学生们买消暑产品。

引用周树人先生的话来向英雄的学生们致意

对，是学生，可学生又是什么？学生，是知识阶级的预备军，并终将成为他的生力军和主力军，对不对。这样刚才那个问题就很清楚了，你们是知识阶级。而知识阶级该是怎样的呢？在我看来，他永远是精神界的战士，他永远不满足现状，永远不图利害，因而永远都处于痛苦，并随时做出牺牲。但同时，他又是独立而清醒的，从不人云亦云，见风使舵，或随波逐流。也不会一窝蜂的充当看客，或虚张声势的跳到台上去做戏。因此他又是孤独的，富于洞察力的，他会从天上看见深渊，真的知识分子因此而获得自己独立的价值，社会的不断进步，正需要这样的永远不满足于现状，永远不合时宜的真的知识阶级。

最后，恳请大家转发视频，让更多的人看到，请大家关注墙墙的微博，让我们一起发光。



青稞大表白墙 ♂

扫描二维码，关注我的微博



再见。

关于若干问题的解释说明

[TOC]

关于黑色晚礼服和民国风有很多咨询我的，我来统一解释一下

简单说明

黑色礼服和民国学生服无法凑齐。黑色礼服现在只有small号码和加大码。

黑色礼服和民国学生服如下所示





原因解释

我们青岛农业大学的摄影团队的用户使用了这两套衣服，并且该学校会使用这两套衣服较长时间。

大家都知道的，咱们的价格本身就是压到了最低，再加上青岛农业大学的团队已经交齐了钱，再这种情况下，我没有办法说服我的老板将这两套衣服留给大家。抱歉

回答大家的问题

我们也是提前预约的？

主要是青农的那个学校，人家交钱了。我这个我也没有办法。我也确实是尽力去争取了。咱这个价格本身就是压到很低了，这么低的价格，我也没有办法去向服装库去做什么过多的要求。

我们也先把钱交上行吗？

这不是交钱不交钱的事儿。是我们老板，他看到有交钱的这个班级，他肯定先紧着交钱的班级去拍。

这个其他的衣服现在已经不是钱不钱的事儿了。是别的那个班级或者说别的学校里边儿已经用上这套衣服了

那怎么办？

现在我拼老命帮大家护住了以下几套号码齐全的衣服

5种jk制服，5种礼服，汉服和男士的西服和汉服。也就是说除了黑色礼服和民国服饰，其他皆可。

我请教了我们的造型师，造型师推荐大家用灰白色的礼服替代，理由如下口

这个礼服很好看而且它很方便穿，你要是弄的那种很不方便，穿的衣服还很浪费时间，我今天我为什么挑这个灰色的，因为

它百搭它你像那个黑色的他还真不一定百搭，你像嘿嘿，这个皮肤黑一点的，或者啥的，穿黑色的就不好看，这个灰色的它显白啊

下面是灰白色礼服











那婚纱呢？

婚纱，一开始就有说明数量稀少，婚纱尚可争取，五六套还可以争取，那种一个班十几个女生的，确实争取不了

可是我就是想要这个黑色晚礼服/民国风

真的很抱歉，这个我真的做不到，幸好大家还没有交钱，很抱歉耽误了大家的时间，如果大家确实喜欢这两套的话，建议大家去换一家，然后为了表达我的歉意，我愿意自费送给大家（要拍黑色晚礼服/民国风，现在因为我们无法提供服装而换别家的同学）每人提供一套海马体风格证件照。

很抱歉耽误您时间了。

如果我选汉服，会好看吗？

这个您放心，下图是我们昨天拍的汉服，是崂山校区的一个宿舍口，还没有修图



最终说明

决定拍的，我们给您替代黑色晚礼服的灰白色礼服来拍。

不拍的，我们赠送您一套海马体风格证件照。

我也是个打工的，不是老板，我只能在我能力范围内给大家提供最好的选择。抱歉。

商品上架格式

商品上架前需要填写的表格

A	B	C	D	E	F	G	H	I
bigname	name	title	desc	tag	price	origin-price	num	thumb
fruit	mangguo	越南精品芒果	芒果，汁水多，皮薄，好吃还便宜	新品	9.99	19.99	1	https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com

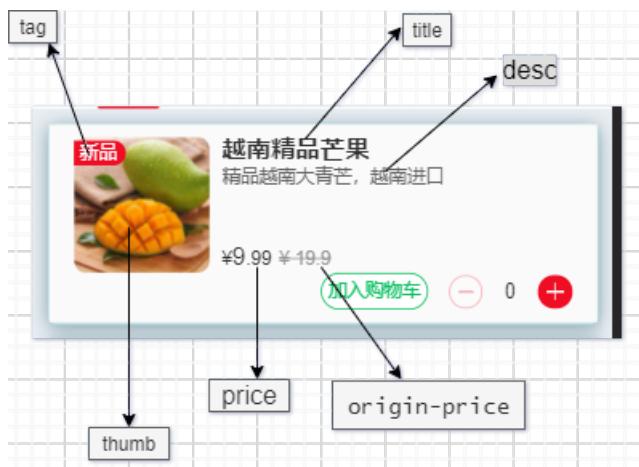
表格内容解释

- **bigname** 商品所属大类，比如，芒果的大类是水果，所以它的**bigname**就是**fruit**。注意：该大类是固定的，比如所有水果商品的大类都是**fruit**，所有糕点类产品的大类都是**gaodian**。现在有三个大类，具体所代表的属性会在后面解释。
- **name** 商品所属小类，建议用商品名称的拼音来代替，小类属于大类的一部分，相同的大类下，不可以有相同**name**的小类。
- **title** 小程序上商品栏的名称，比如“越南精品芒果”，这个限制在7字以内
- **desc** 商品简介
- **tag** 商品的营销标签，选填，比如“爆款”
- **price** 商品的价格，数字，单位是元
- **origin-price** 商品的原价，在小程序之中，原价在价格的下面，并且原价被打×
- **num** 必填 默认为1，商品的原始数量。
- **thumb** 商品的图片链接，比如
<https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif>，获取图片链接的方法是在csdn上上传，然后在浏览器右键图片获取图片链接

小程序中的商品列表



商品列表标注



json格式（从数据库获取的格式）

这个格式是为了让大家理解每一个标签的具体意思



```

fruit: {
  mangguo: {
    title: "越南精品芒果",
    desc: "精品越南大青芒, 越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0
  },
  mangguo1: {
    title: "越南精品芒果",
    desc: "精品越南大青芒, 越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0
  },
  mangguo2: {
    title: "越南精品芒果",
    desc: "精品越南大青芒, 越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0
  },
  mangguo3: {
    title: "越南精品芒果",
    desc: "精品越南大青芒, 越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0
  }
},
gaodian: {
  mangguo: {
    title: "越南精品芒果",
    desc: "精品越南大青芒, 越南进口",
    price: 9.99,
    ...
  }
}

```

```
    tag: "甜品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0

},
mangguo1: {
    title: "越南精品芒果",
    desc: "精品越南大青芒，越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0

},
mangguo2: {
    title: "越南精品芒果",
    desc: "精品越南大青芒，越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0

},
mangguo3: {
    title: "越南精品芒果",
    desc: "精品越南大青芒，越南进口",
    price: 9.99,
    tag: "新品",
    thumb: "https://suyuesheng-biaozhun-blog-tupian.oss-cn-qingdao.aliyuncs.com/blogimg/20210322004059.jfif",
    originprice: 19.9,
    num: 0

}
}
```

附件

[excel表格 商品上架需要填写此表格](<https://www.yuque.com/docs/share/0a3ed3bb-990e-419c-8111-6015d315bd56?#6edd>
《工作簿1》)

如何在印刷品中使用遵循SIL Open Font License协议的字体

昨天在知乎看到了一个问题：[\(如何在设计中声明字体开源许可证？ - 知乎 \(zhihu.com\)\)](#)。恰好最近在研究一些开源协议，所以想要根据原有的协议条款来分析一下如何在印刷品中使用开源字体。我在知乎上面写了一个回答以后感觉有些片面，所以在这里补充一下。

==以下内容，仅供参考，并非法律建议。==

什么是开源字体

先明确一个事情，开源字体指的是字体软件，“字型软件”可以包括源文件(**source files**)、构建脚本(**build script**)以及说明文档。根据我之前的了解，依照美国的现有法律，针对字体的版权是只针对字体软件的，而字体本身往往是被看作社会共有财富的一部分。所谓的字体授权，是围绕在字体软件周围的。在SIL Open Font License里面也明确表示了是针对字体软件(**font software**)。但是，虽然是针对的字体软件，但并不意味着就可以随意使用这个字体，在现在这个时代，大多数印刷品使用字体的方式应该就是在电脑导入字体软件，然后使用该字体然后打印，除非是在印刷品上用手画了一个和原字体一样的字体，不然，还是要遵守协议的。

如何使用

先说结论：印刷品可以使用这个字体并且使用了该字体的印刷品可以商用，印刷品无需因使用这个字体而使用和原字体同样的声明，印刷品无需因使用了**SIL Open Font License**下授权的字体而包含该字体的授权条款和版权声明。用人话说就是“放心大胆的使就行，不用在印刷品里面添加这个字体所使用的授权声明”。但是，印刷品在电脑上的源文件，比如包含字体源文件的`psd`、`word`等，如果使用了该协议授权的字体，那么还是需要在分发源文件的时候声明所使用字体的版权（声明方式可以是印刷品源文件和字体版权协议放在同一个文件夹下）。如果印刷品源文件只是设置使用什么字体而没有在源文件中包含字体软件的任何部分（比如说，我设置使用宋体，但是最终显示的效果要取决于你的电脑上有没有安装这个字体。文字工作者应该很好理解这个和包含字体源文件的区别。）就不用声明字体授权了。

下面是详细解释：

首先，==Source Han Sans使用的是SIL Open Font License。==

==OFL允许以本授权释出的字型自由地使用、研究、修改和再分发(redistributed)，而该释出字型不得被单独销售。==但是使用该字体的软件是可以售卖的，也就是说，使用这个字体的产品是可以商用的。

在SIL Open Font License中针对使用该字体的文档的声明是这样的：

The Font Software, modified or unmodified, in part or in whole, must be distributed entirely under this license, and must not be distributed under any other license. The requirement for fonts to remain under this license does not apply to any document created using the Font Software.

翻译成中文就是：

“字型软件”，无论已修改或未修改、部分或整体，均必须完全通过本授权下分发，不得在任何其他授权条款下分发。本授权针对释出字型“必须以同样授权释出”的要求规定，并不适用于任何使用该“字型软件”创建的任何文档。

也就是说使用该字体的任何文档都不需要强制在这个协议下分发，你的印刷品完全可以不使用这个协议去使用其他的协议或者保留你关于这个印刷品的所有版权。

但是，这只是说，“你的印刷品”在发布的时候无须沿用这个协议，但是既然使用了这个字体，你必须遵守这个字体的协议。关于对这个字体的使用，该协议是这样规定的：

Original or Modified Versions of the Font Software may be bundled, redistributed and/or sold with any software, provided that each copy contains the above copyright notice and this license. These can be included either as stand-alone text files, human-readable headers or in the appropriate machine-readable metadata fields within text or binary files as long as those fields can

be easily viewed by the user.

翻译成中文：

"字型软件"的"原始版本"或"修改版本"可以与任何软件捆绑 (bundled)、再分发以及 / 或一并销售，前提为每份软件副本都必须包含本授权条款上述的版权声明 (copyright notice) 以及本授权条款全文。这些版权声明与条款全文可以被放置在独立纯文本文件、人类可读信息头、或文本 / 二进制文件内适当的、用户易于查阅浏览的机器可读元数据字段。

对标注==所使用字体及字体修改版本==的版权声明要求仅适用于软件 (software)。现在就只有一个问题了，印刷品是否属于软件？这个问题不要想当然，我们看一下有没有相关的内容或者法律条文去指定软件的范围。我通过网络查询了一下，软件最基本的定义就是“软件需有硬件才能运作”。如此看来印刷品绝非软件。

如此看来，印刷品可以使用这个字体并且使用了该字体的印刷品可以商用，印刷品无需因使用这个字体而使用和原字体同样的声明，印刷品无需因使用了**SIL Open Font License**下授权的字体而包含该字体的授权条款和版权声明。用人话说就是“放心大胆的使就行，不用在印刷品里面添加这个字体所使用的授权声明”。

开源软件的一般规律

常见的开源软件一般都是可以自由使用的，这个自由不是免费而是freedom。也就是说，仅限自己使用开源软件而不涉及分发的话，是不用担心授权问题的。如果涉及分发就需要具体问题具体分析了。

LICENSE

copyright © 2021 苏月晟，版权所有。



本作品由苏月晟采用[知识共享署名-非商业性使用-相同方式共享 4.0 国际许可协议](#)进行许可。

移动web端开发之对微信支付文档的自我理解

微信支付开发者文档 ([qq.com](#))

前端开发中微信支付的分类

在移动前端开发中微信支付大概可以分成两部分，分别是网页调用微信支付和小程序调用微信支付，其中网页调用微信支付又可以分为微信内置浏览器(微信公众号)调用微信支付和普通浏览器H5页面调用微信支付。

微信支付开发文档中，对H5支付是这样描述的

H5支付是指商户在微信客户端外的移动端网页展示商品或服务，用户在前述页面确认使用微信支付时，商户发起本服务呼起微信客户端进行支付。

说明：要求商户已有H5商城网站，并且已经过**ICP备案**，即可申请接入。

微信支付中对jsapi支付是这样描述的

JSAPI支付适用于线下场所、公众号场景和**PC网站场景**。

商户已有H5商城网站，用户通过消息或扫描二维码在微信内打开网页时，可以调用微信支付完成下单购买的流程。

也就是说可以有以下这个表格来展示移动前端调用微信支付

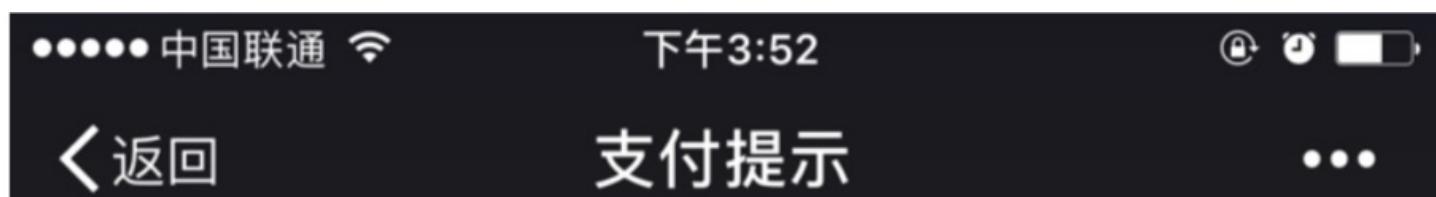
- web调用微信支付
 - 微信内置浏览器外网页调用微信支付使用**H5**专属微信支付文档调用微信支付 [微信支付-开发者文档 \(qq.com\)](#)
 - 微信内置浏览器调用微信支付使用 **JSAPI**调用微信支付 [微信支付-普通下单开发者文档 \(qq.com\)](#)
- 小程序调用微信支付
 - 使用 小程序专属调用微信支付文档开发 [微信支付-开发者文档 \(qq.com\)](#)

移动前端网页调用微信支付面临的两难困境

首先，从技术上来说，H5专用微信支付和**JSAPI**调用微信支付实际上是大同小异，也就是说这两者开发的难度基本一致，开发的过程也大差不差，这两个会一个就能很快地掌握另一个。通过JavaScript来分辨当前浏览器是否是微信内置浏览器也是可以实现的，也就是说一套代码既可以在微信内置浏览器支付也可以在微信以外的浏览器支付在技术上是完全可行的。但是，同样都是网页，却要用两种方式进行支付难免让人不爽！

其实，微信的做法在技术和商业是完全可以理解的，首先，因为调用支付有两种方式，势必会让一部分开发者选择JSAPI进行微信内置浏览器的支付，这样就会把一部分用户的支付行为完全地锁在微信客户端内进行，而不是微信客户端以外的部分调用微信客户端进行支付，其次，微信内置浏览器的支付行为有很大一部分是和微信公众号相联系的，按照正常的小平台的逻辑（比如我现在开发的这个平台），一般的思路是微信公众号打开网页→网页通过微信公众号获取微信用户信息→将微信用户信息转换为当前网页的用户信息，并储存在数据库上面→调用微信支付。这样微信支付势必会发生在也只能发生在微信的内置浏览器上面。

但是，微信内置浏览器的支付又会产生另一个问题，无法调用支付宝支付，是不是似乎能真正地理解微信的“良苦用心”了？也是，微信肯定不允许你用着我微信平台的公众号却用支付宝进行支付这种行为的。当然了，如果就是想让用户用着微信公众号却用支付宝进行支付也不是没有办法，引导用户打开浏览器进行支付宝支付不就行了，就像下面这幅图一样，[浅谈微信内置浏览器调用支付宝支付完整教程《1》CSDN博客](#)，所以，应该根据你的web前端服务对象来选取如何进行调用微信支付。



请在菜单中选择在浏览器中打开，
以完成支付



在Safari中打开

我的python加密方案

前言

首先，**python**是动态语言，解释型语言，边编译边运行的特点就决定了他不太好加密，在现有的加密方案里面，只要用心，绝大部分都能被破解(被别人看到源代码)，这是现实。

知乎这篇文章将现有的几个方案的利弊说的很清楚

如何保护你的 Python 代码（一）——现有加密方案 - Prodesire的文章 - 知乎 <https://zhuanlan.zhihu.com/p/54296517>

那么我们应该怎么办呢？

笔者的建议是，如果代码写的足够好，那就直接开源，开源带给你的声誉和利益可能远高于闭源，这个可以参考Linux kernel。

如果代码写的一般，那么开源的意义就不大了，如果你的代码偏服务类，那么可以采用搭建服务器，提供**web**服务，当然了，这个也不是万能，一是**web**服务几乎相当于把后端接口和接口的使用方法全部告诉了前端，前端代码写的不咋地的话，很容易就被爬虫了，当然，如果后端代码写的不咋地但前端写的很到位，一样会被爬虫。防止爬虫是前端和后端都应该考虑的事情，比如说，数据库应该对核心数据进行一定程度上的加密，就像对密码进行md5加密。再比如说增加ip地址检测机制，再再比如说，前端往后端传数据的时候，蓄意传一些没啥用的假数据，迷惑一下对手，再比如说整些伪元素，数据放css的content里面，想拿数据是吧？先看懂我的js和css再说！或者说修改字符集，123变成321之类的，具体的各大公司的反爬虫手段都在这了
[各大前端巨头反爬虫策略 - 爬虫一只 - 博客园 \(cnblogs.com\)](#)，但是，如果你写的代码压根就不适合提供**web**服务，比如说你卖脚本，那就没办法了，**web**服务不适合你。

既无法开源又无法做**web**服务，还不想用c语言重写核心代码，那就只能想着进行加密了。

我的加密方案

首先，咱要明白咱为什么要加密

1. 防止源代码被别人看到，别人窃取源代码
2. 防止别人虽然看不到源代码或者看不懂源代码，但是已经明白了如何调用你程序的API。

第一步——代码混淆

为了解决上面两个问题，咱们要做的第一件事情就是代码混淆，这一部分的目的就是让你看不懂我的源代码。

第二步——Cython编译核心文件

使用 `Cython` 进行开发的步骤也不复杂。

1) 编写文件 `hello.pyx` 或 `hello.py`:

```
def hello():
    print('hello')
```

2) 编写 `setup.py`:

```
from distutils.core import setup
from Cython.Build import cythonize

setup(name='Hello World app',
      ext_modules=cythonize('hello.pyx'))
```

3) 编译为 `.c`，再进一步编译为 `.so` 或 `.pyd`:

```
python setup.py build_ext --inplace
```

第三步——main.py编写，加入混肴试听的假参数，加入自爆程序

加入我们

我们是谁？



QINGXUAN是杭州三人行电子商务有限公司旗下的品牌。

杭州三人行电子商务有限公司致力于为高校学生提供便捷的校园服务，打造高校学生生活，考试和交友的互联网平台。为高校学生提供闭环生态。

QINGXUAN是为年轻人打造的购物平台。

公司于2020年10月开始进行平台制作和营销思路的创新，2021年1月，取名为“高校团购”于青岛科技大学进行小范围内测。2021年4月进行平台公测。

合作

互联网运营

面向青岛科技大学，中国海洋大学招聘美工一人，文案编辑一人。具体要求如下

职务	工作内容	薪资和股权
美工	1. 制作微信公众号头图 2. 制作海报 3. 制作商品简介所需的各种装饰性图片	200每月，工作三个月以上可获取平台5%的利润
文案编辑	制作微信公众号文案和整理商品简介	200每月，工作三个月以上可获取平台5%的利润

校级总代理

校级代理负责高校的具体营销，工作突出者可以获得公司部分股权，以股东身份加入公司。具体当面详谈。

联系我们

如有意加入我们请联系Email su.yuesheng@foxmail.com 提交简历。

简历内容包括**100**字以内的个人简介和**200**字以内的职业规划和对平台发展的看法。

我们会在24小时之内回复您的邮件。

总结

我们是一个年轻的团队，我们真诚期待你的加入，希望我们可以共同构建高校闭环生态。

加入我们

我们是一家**不谈理想**的创业公司

QINGXUAN 为年轻人提供价格厚道，感动人心的青年良品

日语 时间

变换形式较多的是 4、6、8、9

	年	月	日	時	分
4	よねん	しがつ	よっか	よじ	よんぶん
	四年	四月	四日	四時	四分
6	ろくねん	ろくがつ	むいか	ろくじ	ろっうん
	6 年	6 月	6 日	6 時	6 分
8	はちねん	はちがつ	ようか	はちじ	はっうん
	8 年	8 月	8 日	8 時	8 分
9	きゅうねん	くがつ	ここのか	くじ	きゅうふん
	9 年	9 月	9 日	9 時	9 分

日语 时间的量

变换形式较多的是 4、6、8、9

	年間	か月	日間	時間
4	よねんかん	よんかげつ	よっかかん	よじかん
	四年間	四か月	四日間	四時間
6	ろくねんかん	ろっかげつ	むいかかん	ろくじかん
	六年間	六か月	六日間	六時間
8	はちねんかん	はっかげつ	ようかかん	はちじかん
	八年間	八か月	八日間	八時間
9	きゅうねんかん	きゅうかげつ	ここのかかん	くじかん
	九年間	九か月	九日間	九時間

星期

星期一	月曜日	げつようび
星期二	火曜日	かようび
星期三	水曜日	すいようび

星期一	月曜日	げつようび
星期四	木曜日	もくようび
星期五	金曜日	きんようび
星期六	土曜日	どようび
星期天	日曜日	にちようび

日语学习资源的分享

链接来自互联网，如有侵权，请通过**sys088519@163.com**联系我，我将会删除。本人仅是本着互联网精神进行分享，并没有从中获取任何利润，此篇博文严禁转载，请尊重创作者的权益，本博文中涉及到的所有文件及链接请在下载后的**24**小时之内删除。特此公告

XDF, 叶子, 葱花, 唐盾.....各种名师应有尽有

新标日, 大家的日语, 新编日语各种教材的教学视频应有尽有

各种教材的PDF, 日语能力考试真题和专项训练的也是应有尽有

基本上, 互联网上能找到的资源都有, 但是, 学习日语并不是有了资源就可以了.....

真心建议大家报个线下班或者跟个大学老师先把N3考出来, 这样至少会明白日语的一些基本的知识, 这样自学起来才会事半功倍。

按教材分类

教材名称	老师	网盘链接
《新标日》	XDF王晶	链接: https://pan.baidu.com/s/1ikR2vnW_NGtsr2nRfbICkA 提取码: 5uy0 --来自百度网盘超级会员V1的分享
《新标日》	叶子	链接: https://pan.baidu.com/s/1WGQEyNp224RZYbBCTmwuGg 提取码: xzv3 --来自百度网盘超级会员V1的分享
《新标日》	hj	

说明

一 照片、mv、和底片

- 底片已经在群里发送完毕，平均每班都在100张以上，某班已经协商9号发底片
- 之前说的非常好看的图片，就是摄影师精挑版照片，已经发给每个班的负责人。
- mv的素材已经上传到群里
- 精修图全部发放到位，对精修图有意见的，将照片上传到这个网盘，我们会处理，进行重修，**请在24小时之内上传，重修图片以上传图片为准，重修图片以上传图片为准，重修图片以上传图片为准**



手机打开微信、QQ 扫一扫

- mv 链接：https://pan.baidu.com/s/1Im_DMls4cX3w5S1x_5u71w 提取码：cnfk --来自百度网盘超级会员V1的分享
- 杂志风格图片，**请自取**



二，关于对我们的辱骂和误解

- 要举报我们，请你抓紧去举报
- 说我们扰乱市场？不好意思，根据调查和不完全统计，我们是青岛科大四方校区唯一一个有营业资质且按时交税的工作室。我们的每一个软件，每一个字体，都有来自官方授权。这个市场（青岛科大四方校区），几乎只有我们是合法的。
- 说我们的衣服？如果你不喜欢我们的衣服，当时你就该在拍摄时见到衣服的瞬间，就拒绝拍摄。而不是穿上衣服，拍完了，才来说我们。
- 说无人机？你们学校飞不起来，拍的时候，我们也告诉了你们，当时你们不拒绝我们的拍摄，现在却拿这个来说事情，有意思嘛？当时怎么不拒绝我们？
- 群里的一个同行，别带节奏了。赶紧去税务部门举报我们吧。敢不敢请工商部门来调查一下我们俩？
- 骂我们的，请实名，请起诉我们，请报警。躲在手机后面骂，没意思。我们愿意采取法律手段来维护我们的权益。

前言

毫无疑问，贝多芬在音乐上的造诣是无与伦比的，他是一个天才的音乐家。失聪，一个可以摧毁所有热爱音乐的人的噩梦，却造就了他在后人眼中坚毅不拔的品格。《英雄交响曲》从献给“拿破仑·波拿巴”到献给“英雄”的故事让他蒙上了浓厚的人文主义色彩。渐渐的，人们在谈及贝多芬的时候，除了赞叹他那令人着迷的音乐，也会不由自主地敬佩他经历挫折却不向命运低头的一生。贝多芬不只是活在其他音乐家对他的作品一遍又一遍的虔诚地演奏中，他还活在每一个失意的人心里，“我要扼住生命的咽喉”激励了无数被命运无情嘲弄的人。

贝多芬永生着。

当褪去历史的迷雾，试图去还原一个真实的贝多芬时，人们会发现贝多芬被过多地修饰和包装，他已经从一个具象的人变成了一个有神性的人。本文试图在讲述贝多芬的同时去还原一个真实的天才音乐家——贝多芬。

音乐熏陶

还没有学会告别，就已经后会无期。又是一年毕业季，这是一个流行离开的时代，但我们都擅长告别。你总说毕业遥遥无期，转眼就各奔东西。墙墙青选校园推出了校园留影计划，希望在这最后的日子里，同学们能和自己的舍友、同学、闺蜜、伴侣在快门按下的瞬间留下最美好的回忆。关注青选校园回复“毕业照”即可校园超低价25元/人拍摄毕业照 让我们一起，快门按下，暂停青春。毕业了，大家有什么想说的，快来评论区留言吧！

毕业照拍摄，不要犹豫了。关注青选校园回复“毕业照”即可校园超低价25元/人拍摄毕业照。

什么是雷电接口

Thunderbolt（又称“雷电”，苹果中国译为“雷雳”^[4]）是由[英特尔](#)发表的连接器标准，目的在于当作电脑与其他设备之间的通用总线，第一代与第二代接口是与[Mini DisplayPort](#)集成，较新的第三代开始改为与[USB Type-C](#)结合，并能提供电源。

来源 [维基百科](#)

我们现在常说的雷电接口指的是雷电三接口

第三代改为使用[USB Type-C](#)接口。由于二合一的集成特点，因此它既能以双向 [40 Gbit/s](#) 传输数据（[40 Gbit/s + 40 Gbit/s](#)，特别是针对外置高速网络时），既能兼容[Mini DisplayPort](#)设备直接连接Thunderbolt接口传输视频与声音信号，也可连接[Apple Thunderbolt Display](#)直接同时输出视频、声音与数据，且不用如传统使用多条连接线。

来源 [维基百科](#)

简单来说，能充电，能传输数据且传输速度比较快，能用一条线去输出视频和声音。[c口](#)

可以同时外接多个显示器和拓展多个高性能显卡

青岛科技大学新生报考参考（一）

以下是常规内容，看看就好。

青岛科技大学简介

青岛科技大学（Qingdao University of Science and Technology）[1]位于山东省青岛市，是原化学工业部直属重点高校[2]，入选国家“111计划”[3]，教育部与中国科学院“科教结合协同育人行动计划”高校[4-5]，是国家首批“卓越工程师教育培养计划”、“新工科研究与实践项目”入选高校，[6]是国内第一家设立橡胶专业的高等院校[7]，亚洲唯一的橡胶专业领域高校[8]。曾先后隶属国家轻工业部、国家化学工业部，现为山东省属重点建设的大学和山东省应用基础型人才培养特色名校，被教育部评估为“本科教学工作水平评估优秀高校”和“全国毕业生就业典型经验高校”，被社会赞誉为“中国橡胶工业的黄埔”，CDIO工程教育联盟成员单位。[9]

青岛科技大学各学院专业一览

青岛科技大学2021年选考科目要求				
层次	校区	学院	专业	选考科目要求
本科		法学院	法学	不提科目要求
本科			社会工作	不提科目要求
本科		外国语学院	朝鲜语	不提科目要求
本科			德语	不提科目要求
本科			日语	不提科目要求
本科			俄语	不提科目要求
本科			英语	不提科目要求
本科		传媒学院	汉语言文学	不提科目要求
本科			广告学	不提科目要求
本科			编辑出版学	不提科目要求
本科			动画	不提科目要求
本科			动画(中英合作办学)	不提科目要求
本科		艺术学院	绘画	不提科目要求
本科			服装与服饰设计	不提科目要求
本科			公共艺术	不提科目要求
本科			产品设计	不提科目要求
本科			环境设计	不提科目要求
本科			视觉传达设计	不提科目要求
本科			音乐表演（声乐方向）	不提科目要求
本科			音乐表演（钢琴方向）	不提科目要求
本科			音乐表演（器乐方向）	不提科目要求
本科			数学与应用数学	物理(1门科目考生必须选考方可报考)
本科		数理学院	应用物理学	物理(1门科目考生必须选考方可报考)
本科			应用统计学（校企合作）	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科			机械工程	物理(1门科目考生必须选考方可报考)
本科			机械工程（高分子加工机械方向）	物理(1门科目考生必须选考方可报考)
本科			机械工程（英才计划）	物理(1门科目考生必须选考方可报考)
本科			材料成型及控制工程	物理(1门科目考生必须选考方可报考)
本科			能源与动力工程	物理(1门科目考生必须选考方可报考)
本科			过程装备与控制工程	物理(1门科目考生必须选考方可报考)

本科	机电工程学院 崂山校区	智能制造工程	物理(1门科目考生必须选考方可报考)
本科		新能源科学与工程	物理(1门科目考生必须选考方可报考)
本科		船舶与海洋工程	物理(1门科目考生必须选考方可报考)
本科		油气储运工程	物理(1门科目考生必须选考方可报考)
本科		工业设计	物理(1门科目考生必须选考方可报考)
本科		机械工程(中韩合作办学)	物理(1门科目考生必须选考方可报考)
本科		材料成型及控制工程(中韩合作办学)	物理(1门科目考生必须选考方可报考)
本科	自动化学院	智能电网信息工程	物理(1门科目考生必须选考方可报考)
本科		电气工程及其自动化	物理(1门科目考生必须选考方可报考)
本科		电子信息科学与技术	物理(1门科目考生必须选考方可报考)
本科		测控技术与仪器	物理(1门科目考生必须选考方可报考)
本科		自动化	物理(1门科目考生必须选考方可报考)
本科		机器人工程	物理(1门科目考生必须选考方可报考)
本科		自动化(中法合作办学)	物理(1门科目考生必须选考方可报考)
本科	经管学院	财务管理	不提科目要求
本科		市场营销	不提科目要求
本科		工商管理	不提科目要求
本科		物流管理	不提科目要求
本科		城市管理	不提科目要求
本科		工业工程	不提科目要求
本科		国际经济与贸易	不提科目要求
本科	信息学院	计算机科学与技术	物理(1门科目考生必须选考方可报考)
本科		通信工程	物理(1门科目考生必须选考方可报考)
本科		信息工程	物理(1门科目考生必须选考方可报考)
本科		集成电路设计与集成系统	物理(1门科目考生必须选考方可报考)
本科		微电子科学与工程	物理(1门科目考生必须选考方可报考)
本科		人工智能	物理(1门科目考生必须选考方可报考)
本科		软件工程(校企合作)	物理(1门科目考生必须选考方可报考)
本科		物联网工程(校企合作)	物理(1门科目考生必须选考方可报考)
本科	大数据学院	数据科学与大数据技术	物理(1门科目考生必须选考方可报考)
本科	体育学院	休闲体育	不提科目要求
本科	中德科技学院	自动化(中德合作办学)	物理(1门科目考生必须选考方可报考)
本科		机械工程(中德合作办学)	物理(1门科目考生必须选考方可报考)
本科		应用化学(中德合作办学)	化学(1门科目考生必须选考方可报考)
本科	高分子学院	高分子材料与工程(橡胶方向)	化学(1门科目考生必须选考方可报考)
本科		高分子材料与工程(塑料方向)	化学(1门科目考生必须选考方可报考)
本科		高分子材料与工程(合成方向)	化学(1门科目考生必须选考方可报考)
本科		高分子材料与工程(英才计划)	化学(1门科目考生必须选考方可报考)
本科		功能材料	化学(1门科目考生必须选考方可报考)
本科		复合材料与工程	化学(1门科目考生必须选考方可报考)
本科		包装工程	物理, 化学(2门科目考生选考其中一门即可报考)
本科		高分子材料与工程(中韩合作办学)	化学(1门科目考生必须选考方可报考)
本科		化学	物理, 化学(2门科目考生选考其中一门即可报考)
本科		化学(英才计划)	物理, 化学(2门科目考生选考其中一门即可报考)

本科		化学院 四方校区	应用化学	物理, 化学(2门科目考生选考其中一门即可报考)
本科			应用化学(英才计划)	物理, 化学(2门科目考生选考其中一门即可报考)
本科			分子科学与工程	物理, 化学(2门科目考生选考其中一门即可报考)
本科			应用化学(中美合作办学)	化学(1门科目考生必须选考方可报考)
本科		化工学院	化学工程与工艺	化学(1门科目考生必须选考方可报考)
本科			化学工程与工艺(精细方向)	化学(1门科目考生必须选考方可报考)
本科			化学工程与工艺(英才计划)	化学(1门科目考生必须选考方可报考)
本科			药物制剂	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科			制药工程	化学, 生物(2门科目考生选考其中一门即可报考)
本科		材料学院	材料物理	物理(1门科目考生必须选考方可报考)
本科			材料化学	化学(1门科目考生必须选考方可报考)
本科			金属材料工程	化学(1门科目考生必须选考方可报考)
本科			无机非金属材料工程	化学(1门科目考生必须选考方可报考)
本科			新能源材料与器件(英才计划)	物理(1门科目考生必须选考方可报考)
本科		环境学院	环境工程	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科			环境科学	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科			安全工程	物理(1门科目考生必须选考方可报考)
本科		海洋学院	海洋科学	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科			轻化工程	物理, 化学(2门科目考生选考其中一门即可报考)
本科			生物工程	化学, 生物(2门科目考生选考其中一门即可报考)
本科			食品质量与安全	物理, 化学, 生物(3门科目考生选考其中一门即可报考)
本科	中德生态园校	中德工程学 区	复合材料与工程(中德合作办学)	化学(1门科目考生必须选考方可报考)
本科			化学工程与工艺(中德合作办学)	化学(1门科目考生必须选考方可报考)
本科		高密校区	英语	不提科目要求
本科			国际经济与贸易(校企合作)	不提科目要求
本科			化学工程与工艺	化学(1门科目考生必须选考方可报考)
本科			高分子材料与工程	化学(1门科目考生必须选考方可报考)
本科			计算机科学与技术	物理(1门科目考生必须选考方可报考)
本科			机械工程	物理(1门科目考生必须选考方可报考)
本科			电气工程及其自动化	物理(1门科目考生必须选考方可报考)
专科			商务英语	不提科目要求
专科			财务管理	不提科目要求
专科			市场营销(校企合作)	不提科目要求
专科			产品艺术设计	不提科目要求
专科			环境艺术设计	不提科目要求
专科			视觉传播设计与制作	不提科目要求
专科			橡胶工程技术	化学(1门科目考生必须选考方可报考)
专科			高分子材料工程技术	化学(1门科目考生必须选考方可报考)
专科			应用化工技术	化学(1门科目考生必须选考方可报考)
专科			模具设计与制造	物理(1门科目考生必须选考方可报考)
专科			机械设计与制造	物理(1门科目考生必须选考方可报考)

专科		软件技术	物理(1门科目考生必须选考方可报考)
专科		工业过程自动化技术	物理(1门科目考生必须选考方可报考)
专科		电气自动化技术	物理(1门科目考生必须选考方可报考)
专科	中德生态园校区	商务英语	不提科目要求
专科		工业过程自动化技术	物理(1门科目考生必须选考方可报考)
专科		高分子材料工程技术	化学(1门科目考生必须选考方可报考)
专科		计算机应用技术	物理(1门科目考生必须选考方可报考)

Namespace ConsoleApp1

Classes

[BookA](#)

用于测试的类

[DefaultFun](#)

常规意义上为实例化对象准备方法

[JsonDocumentExtensions](#)

对json进行操作的拓展方法

[StringExtensions](#)

为string类型增加拓展方法

[TestA](#)

Delegates

[WTuo](#)

Class BookA

用于测试的类

Inheritance

System.Object

BookA

Inherited Members

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.GetType()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public class BookA
```

Constructors

BookA()

Declaration

```
public BookA()
```

BookA(String, Int32, String)

定义书的基本信息

Declaration

```
public BookA(string name, int peices, string outCompany = "无")
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	书名
System.Int32	peices	价格
System.String	outCompany	出版商

Fields

timestamp

时间戳

Declaration

```
[JsonInclude]  
public long timestamp
```

Field Value

TYPE	DESCRIPTION
System.Int64	

Properties

Author

作者

Declaration

```
[JsonPropertyName("作者")]  
public string Author { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

defaultFun

Declaration

```
public DefaultFun defaultFun { get; set; }
```

Property Value

TYPE	DESCRIPTION
DefaultFun	

Details

Declaration

```
[JsonInclude]  
public Dictionary<string, int> Details { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Int32>	

ExtensionData

储存反序列化时候的溢出数据

Declaration

```
[JsonExtensionData]  
public Dictionary<string, JsonElement> ExtensionData { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Text.Json.JsonElement>	

Name

书的名称

Declaration

```
[JsonInclude]
public string Name { set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

OutCompany

书的出版商

Declaration

```
[JsonIgnore]
public string OutCompany { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Time

出版时间

Declaration

```
public DateTime Time { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.DateTime	

Methods

Finalize()

析构函数（终结器）

Declaration

```
protected void Finalize()
```

One()

overload

Declaration

```
protected string One()
```

Returns

TYPE	DESCRIPTION
System.String	

One(Int32)

Declaration

```
protected string One(int a)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	a	

Returns

TYPE	DESCRIPTION
System.String	

One(String)

Declaration

```
protected void One(string str)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	str	

ToString()

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()

Class DefaultFun

常规意义上为实例化对象准备方法

Inheritance

System.Object

DefaultFun

Inherited Members

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.GetType()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

System.Object.ToString()

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public class DefaultFun
```

Properties

Name

Declaration

```
public string Name { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Class JsonDocumentExtensions

对json进行操作的拓展方法

Inheritance

System.Object

JsonDocumentExtensions

Inherited Members

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.GetType()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

System.Object.ToString()

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public static class JsonDocumentExtensions
```

Class StringExtensions

为string类型增加拓展方法

Inheritance

System.Object

StringExtensions

Inherited Members

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.GetType()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

System.Object.ToString()

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public static class StringExtensions
```

Class TestA

Inheritance

System.Object

TestA

Inherited Members

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.GetType()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

System.Object.ToString()

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public class TestA
```

Methods

GetPropertyValue<T>(T)

获取对象的属性和值

Declaration

```
public static Dictionary<string, string> GetPropertyValue<T>(T obj)
```

Parameters

TYPE	NAME	DESCRIPTION
T	obj	对象

Returns

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.String>	返回属性与值一一对应的字典

Type Parameters

NAME	DESCRIPTION
T	

Delegate WTuo

Namespace: [ConsoleApp1](#)

Assembly: ConsoleApp1.dll

Syntax

```
public delegate void WTuo();
```