

# p4版本管理工具使用

## 1. p4 sync

下载, 更新.

- \* 更新全部 `p4 sync`
- \* 更新指定文件 `p4 sync //Acme/dev/jam/Jamfile`
- \* 更新指定文件指定版本 `p4 sync //Acme/dev/jam/Jamfile#1`
- \* 更新指定目录 `p4 sync //Acme/dev/jam/...`

一般非stream下载方式

```
mkdir 11111
cd 11111
setenv P4CLIENT xxxxx
p4 client; # then edit the view by yourself
p4 sync
```

## 2. p4 add

添加文件到p4 changelist, 然后通过p4 submit命令就可以上传到p4

这个跟git有点类似。

## 3. p4 edit

将文件放入p4 changelist, 置为open状态。只有放入到changelist里面的文件才会在p4 submit的时候上传到p4库。可以理解是git add命令。

## 4. p4 revert

将已经放入p4 changelist,但是还没有p4 submit的文件, 回退到修改前的版本。

```
p4 revert xxxx_file
```

## 5. p4 diff

只有放入p4 changelist里面的文件 (即需要先用p4 edit xxx命令), 才可以进行p4 diff

这跟git不一样,git可以直接git diff, p4 有些麻烦。

## 6. p4 print

将p4库上指定文件，指定版本的内容打印出来，可以使用>号打印到一个文件，比如：

```
p4 print //p4_depot_xxx/file_xxx#20 > 1
```

这样可以用gvimdiff命令，执行本地文件与指定p4库版本的文件进行比较。

## 7. p4 submit

将p4 changelist里面的文件上传到p4

```
p4 submit -d "xxxx"
```

## 8. p4 changelist

显示或编辑进入p4 changelist的文件，我本人一般只是用p4 changelist -o命令来显示changelist内容。

## 9. p4 have

查看本地文件版本

```
p4 have Jamfile
```

## 10. p4 label

打tag

## 11. p4 labels

Display list of defined labels. 它是显示全部的tag[] 如果需要指定人打包的可以加上-u xxx参数

示例：

```
p4 labels -u xxx | grep tag_xxx
```

## 12. p4 filelog

查看指定文件的历史版本信息，注意这里显示是的直到p4库上最新的修改，不是到当前目录p4 client版本

为时间截止的修改历史。

```
p4 filelog xxxx  
p4 filelog //p4xxxx/xxxx
```

## 13. p4 changes

查看最近提交的版本list log

```
#查看某库近10次版本提交log  
p4 changes -s submitted -m 10 //p4_depot/xxxx/....  
  
#等价于  
p4 changes -m 10 //p4_depot/xxxx/....
```

## 14. p4 delete

p4 删除文件。