p4版本管理工具使用

1. 区分某个路径是不是stream

```
p4 streams //MyDepot/...
#如果能列出来→是 stream，否则就不是
```

2. 下载

2.1 如果是stream

```
mkpwa -s //stream    本地目录
cd 本地目录

  * 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/...
```

2.2 如果不是stream

```
mkdir 11111
cd 11111
#先setenv P4CLIENT xxxx

# p4 client 编辑路径

# p4 sync
```

3. populate

从p4库上将一个目录copy到另外一个目录

3.1 普通目录，不是stream

```
#先预演
p4 populate -n //src/main/... //src/release/...

#确认无误，正式执行
p4 populate //src/main/... //src/release/...
```

```
# 下载文件
p4 sync

# 提交上传
p4 submit

# 先查看其它正常stream里的信息
p4 client -o
```

将当前stream的信息补充到当前p4 client里面，保存，就可以p4 submit了。

3.2 是stream

也可以直接通过图形界面操作。

3.2.1 没有父流

```
p4 stream -t mainline //MyDepot/main # 弹出的窗口直接保存即可
p4 populate //别的depot/trunk/... //MyDepot/main/...     # 完成后即可在p4v图形界面上看到stream
```

3.2.2 从父流填充子流

```
# 先创建子 stream
p4 stream -t development -P //proj/main //proj/dev

# populate 填充 -S populate 方向是“父 → 子”，所以加 -r
p4 populate -d "populate dev stream from main" -S //proj/dev -r
#等价于 p4 copy -r -S //MyDepot/dev + 自动 submit
```

4. 撤销提交

已经进changelist, 如果没有进changelist, 直接p4 revert xxx_file即可。

需求	命令	适用场景
待提交，扔掉改动	p4 revert -c CL	本地没提交，不要了
待提交，先存起来	p4 shelve -c CL	临时保存、换机器改
已提交，安全回退（推荐）	p4 undo @CL + submit	主干、多人协作 EDA 项目

```
# 撤销指定CL里所有打开文件，变回未checkout
p4 revert -c 12345 //...
# -n 预览，先看会撤销哪些文件，安全优先
p4 revert -n -c 12345 //...
# 单独revert stream spec
p4 revert -So -c 12345
```

```
# 查看CL内容，确认没有任何条目（无文件、无stream）
p4 opened -c 12345
# 现在可以删除
p4 change -d 12345
```

5. 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。

6. 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。

7. 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。

8. p4 files

显示p4指示路径有哪些文件。

```
p4 files //depot/... Provides information about all files in the depot.
p4 files //clientname/... Provides information about all depot files visible
through the client view.
p4 files @2011/03/10 Provides information about all depot file revisions
that existed on March 10, 2011.
p4 files @2011/03/31:08:00,@2011/03/31:17:00 Lists all files and revisions
changed during business hours on March 31, 2011.
p4 files //depot/proj2/...@p2lab Lists files and revisions under the
directory //depot/proj2/... tagged by label p2lab.
p4 files //depot/file.c Show information on the head revision of
//depot/file.c. (that is, the highest revision in the implied range of
#1,#head)
p4 files -a //depot/file.c Show information on every revision of
//depot/file.c (that is, all revisions in the implied range of #1,#head)
p4 files -A //arch/depot/proj/... If an administrator has used p4 archive to
transfer //depot/proj/... to an archive P4 Command depot named arch,
displays information about the files in the archived project.
```

9. 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。

10. p4 submit

将p4 changelist里面的文件上传到p4

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

11. 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。

12. p4 have

查看本地文件版本

```
p4 have Jamfile
```

13. p4 label

打tag

14. 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
```

15. 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
```

16. 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/....
```

查看有哪些pengding changes

```
p4 changes -s pending -u xxx
```

删除对应p4 change

```
p4 change -d change_number
```

17. p4 delete

p4 删除文件。

18. p4 client

删除p4 client

```
p4 client -d clientname
```