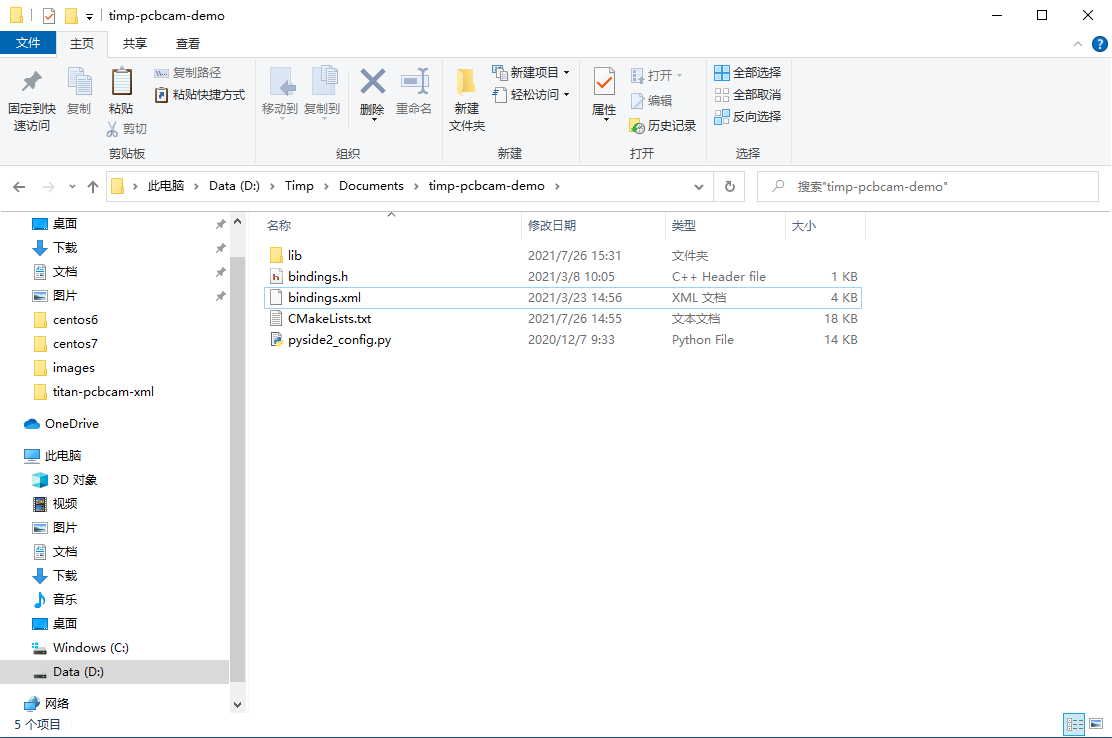
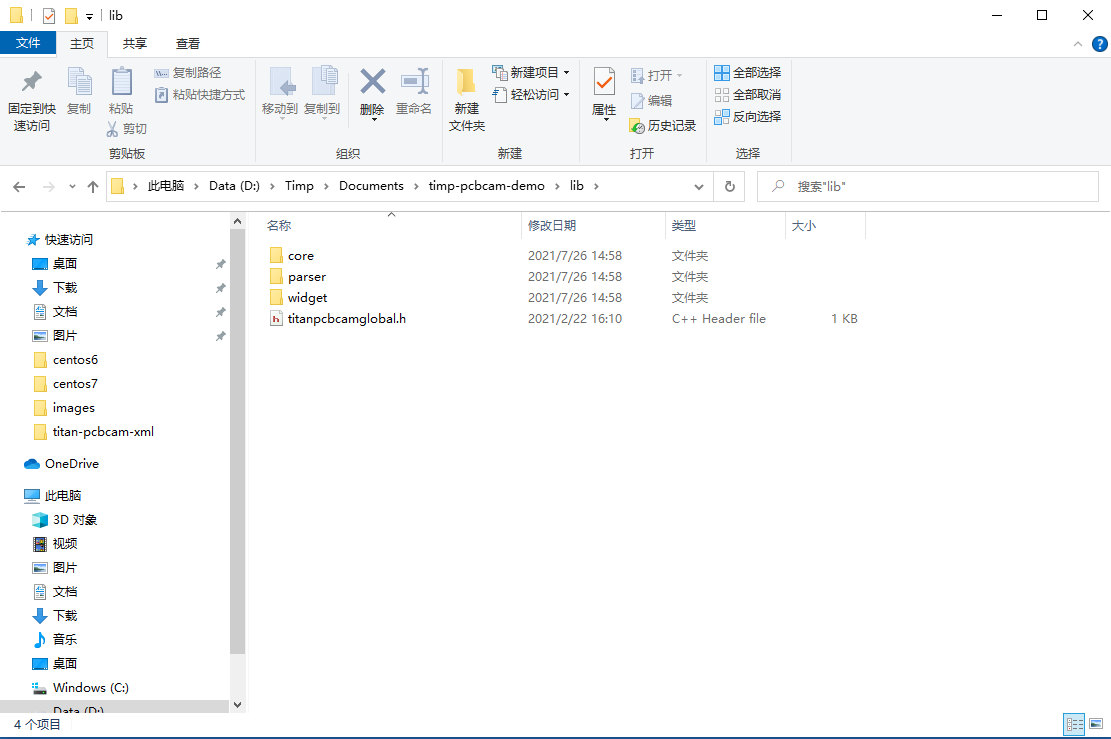
如何在windows系统下使用shiboken制作一个python扩展库

提醒：为避免出错，建议复制本例所用到的配置文件做修改

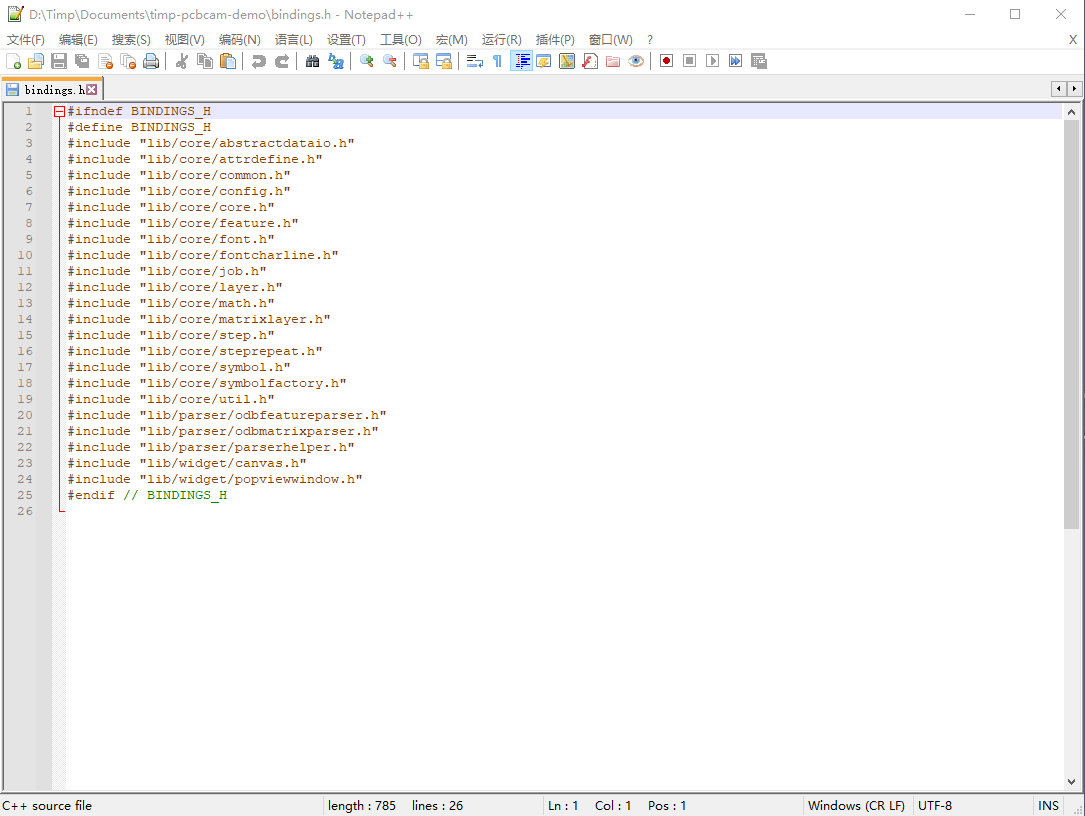
Demo目录结构



1. 新建文件夹，如：timp-pcbcam-demo
2. 在 timp-pcbcam-demo 下新建lib文件夹，存放c++源码

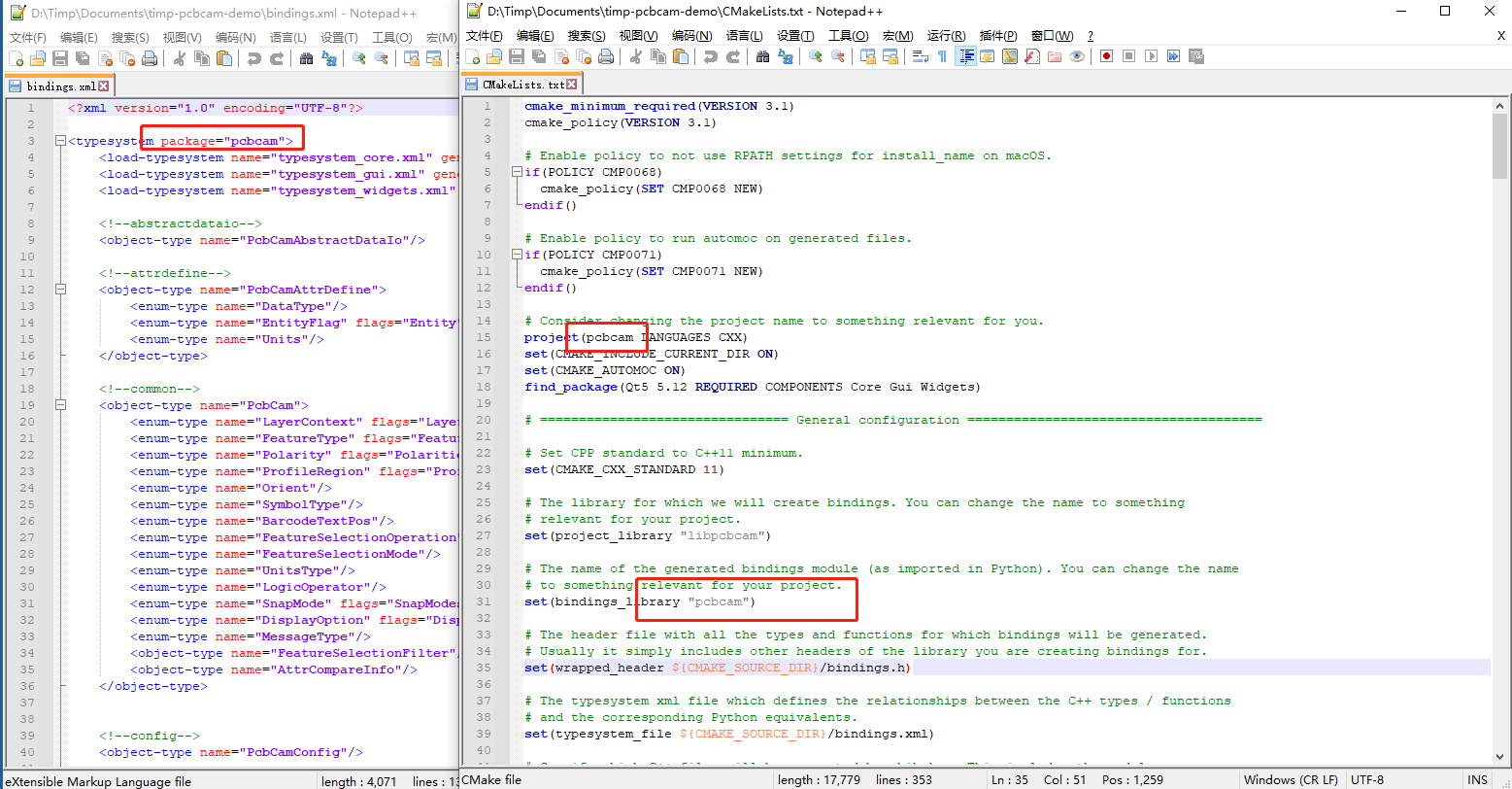


1. 在 timp-pcbcam-demo 下新建bindings.h文件, include所有需要转换代码的头文件

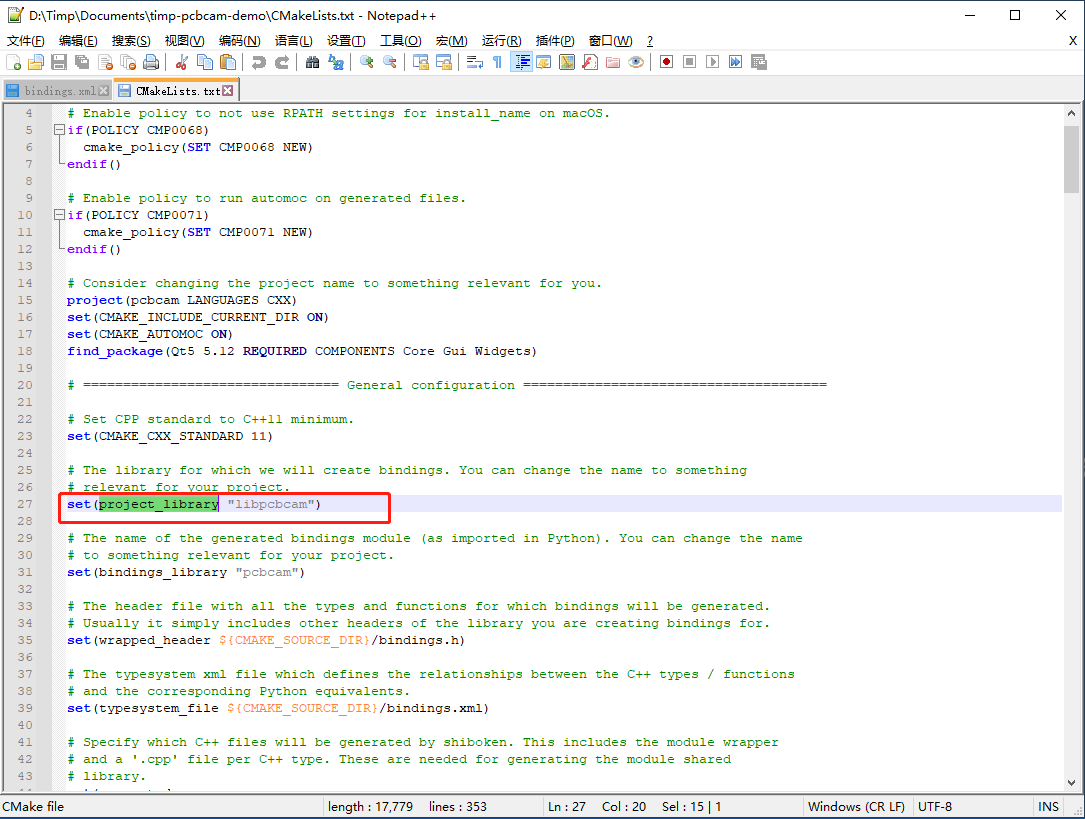


1. 在 timp-pcbcam-demo 下新建bindings.xml文件, 根据typesystem语法文档编写xml，<https://doc.qt.io/qtforpython-6/shiboken6/typesystem.html>
2. 拷贝pyside2\_config.py文件至 timp-pcbcam-demo 下
3. 拷贝CMakeLists.txt至 timp-pcbcam-demo 下并修改

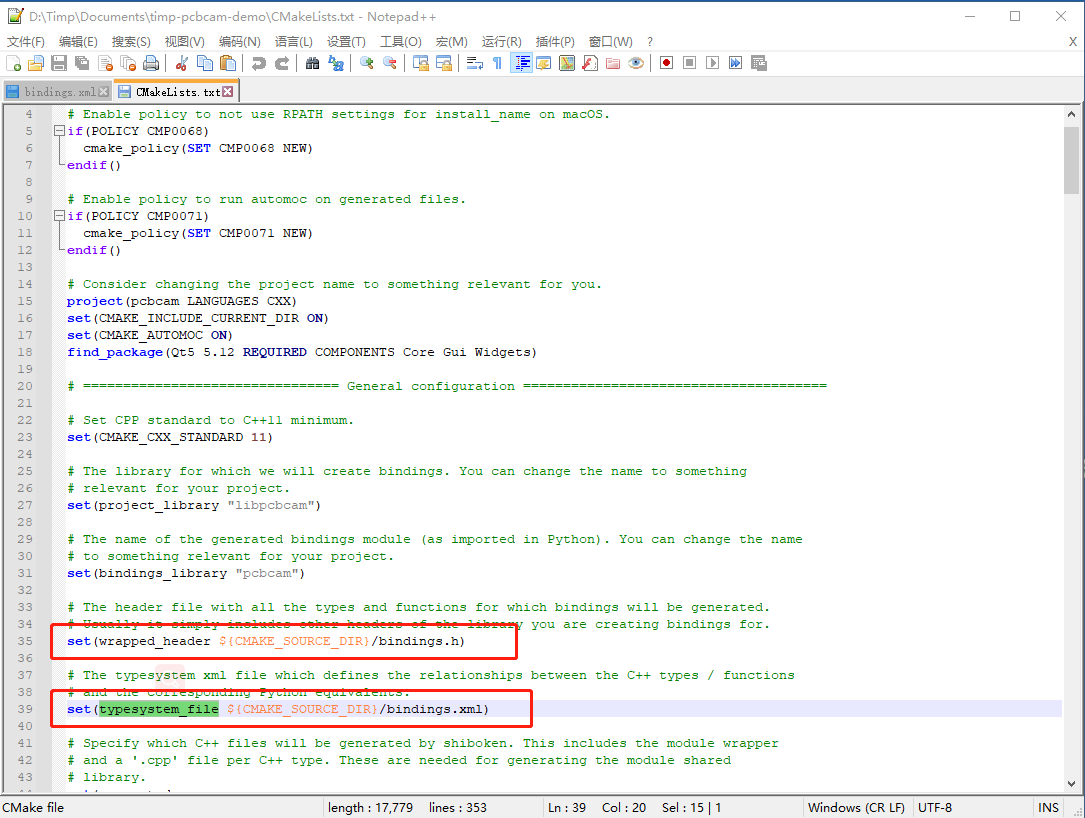
* 修改project名称与bindings\_library名称，注意bindings\_library需要与bindings.xml的package名称保持一致



* 修改project\_library名称，此处是生成的动态库的名称

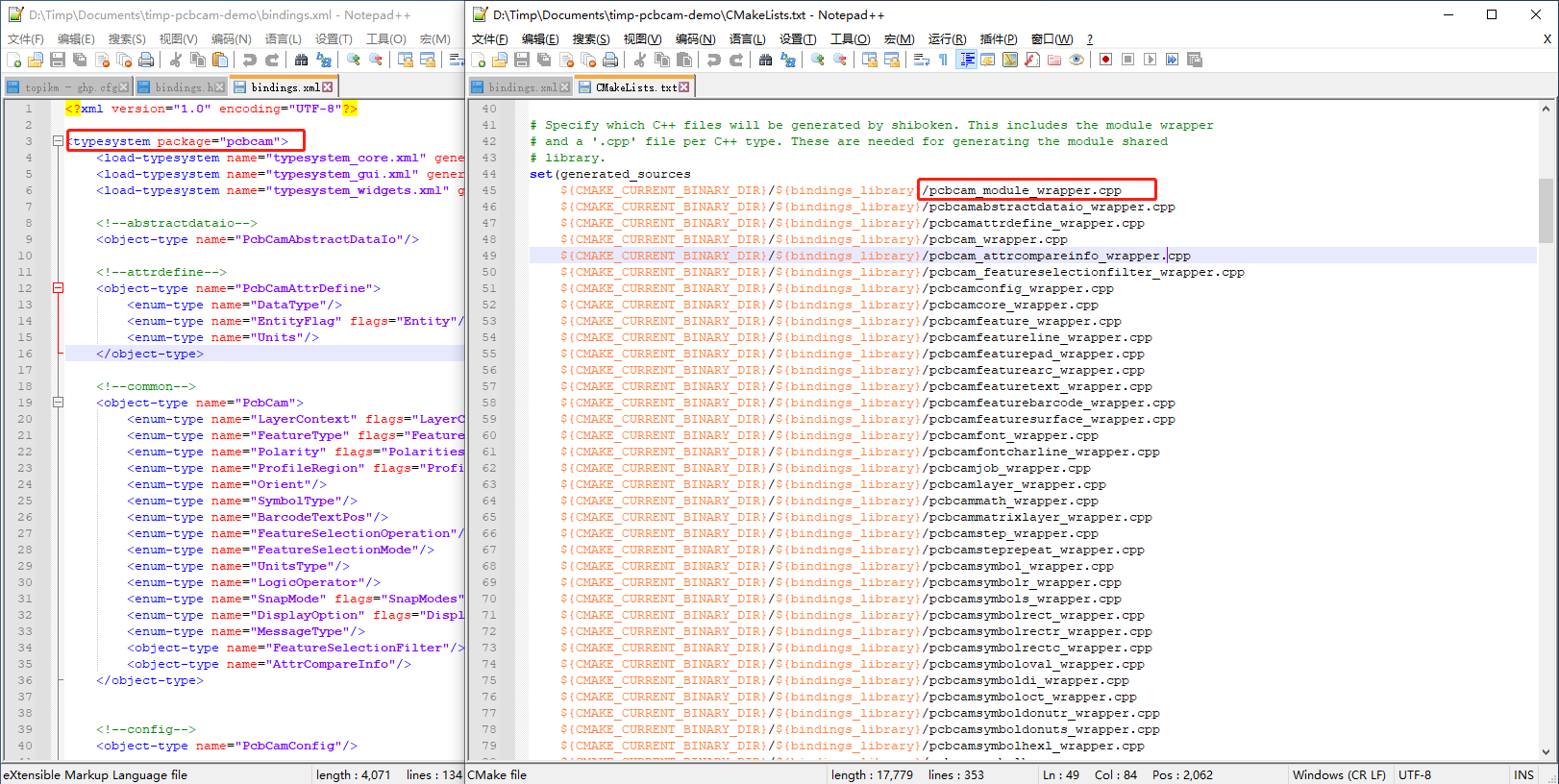


* 指定wrapped\_header和typesystem\_file

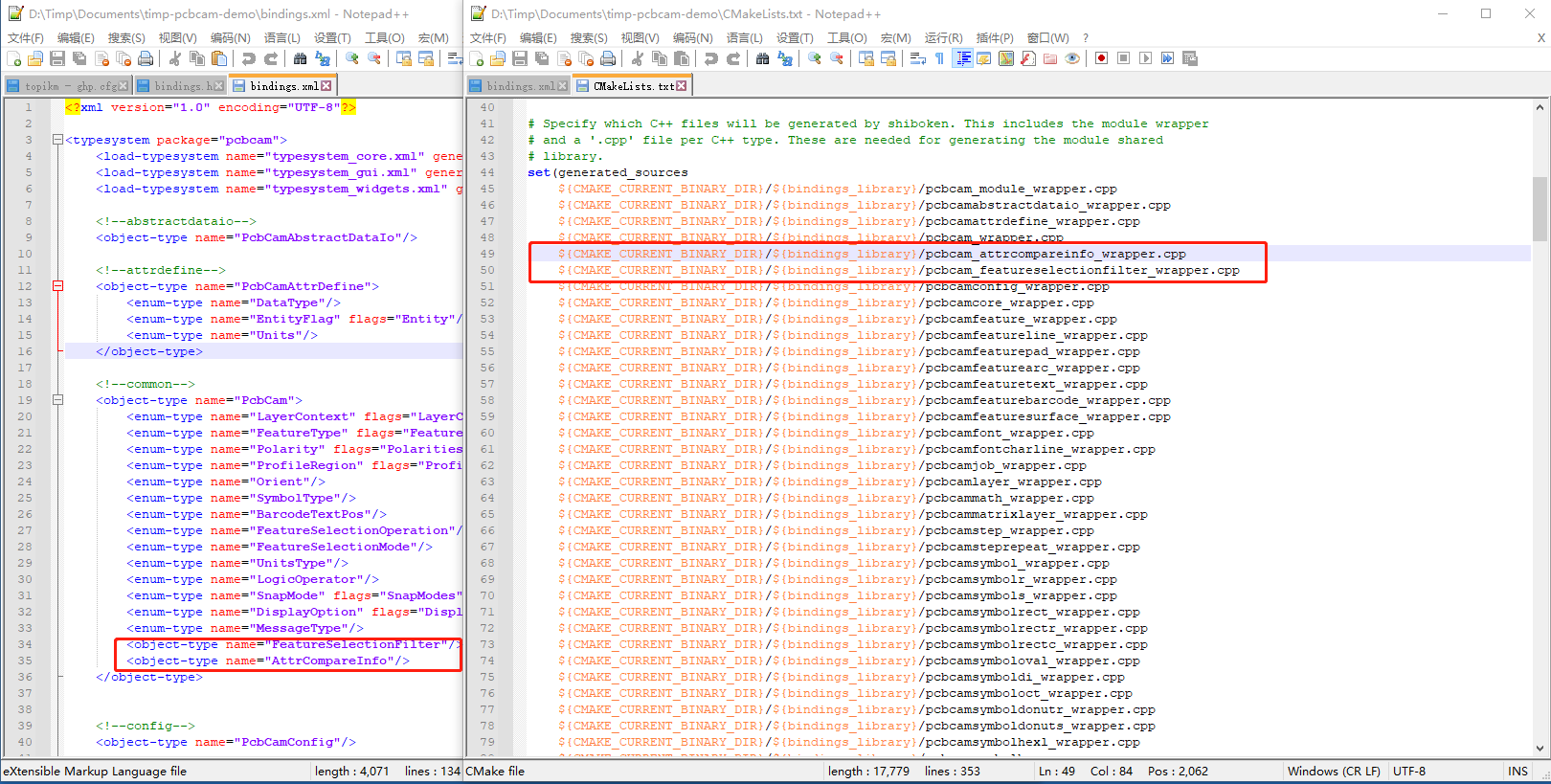


* 指定generated\_sources，此处需要指定所有需要转换的c++类，也就是bindings.xml中所有object-type节点。

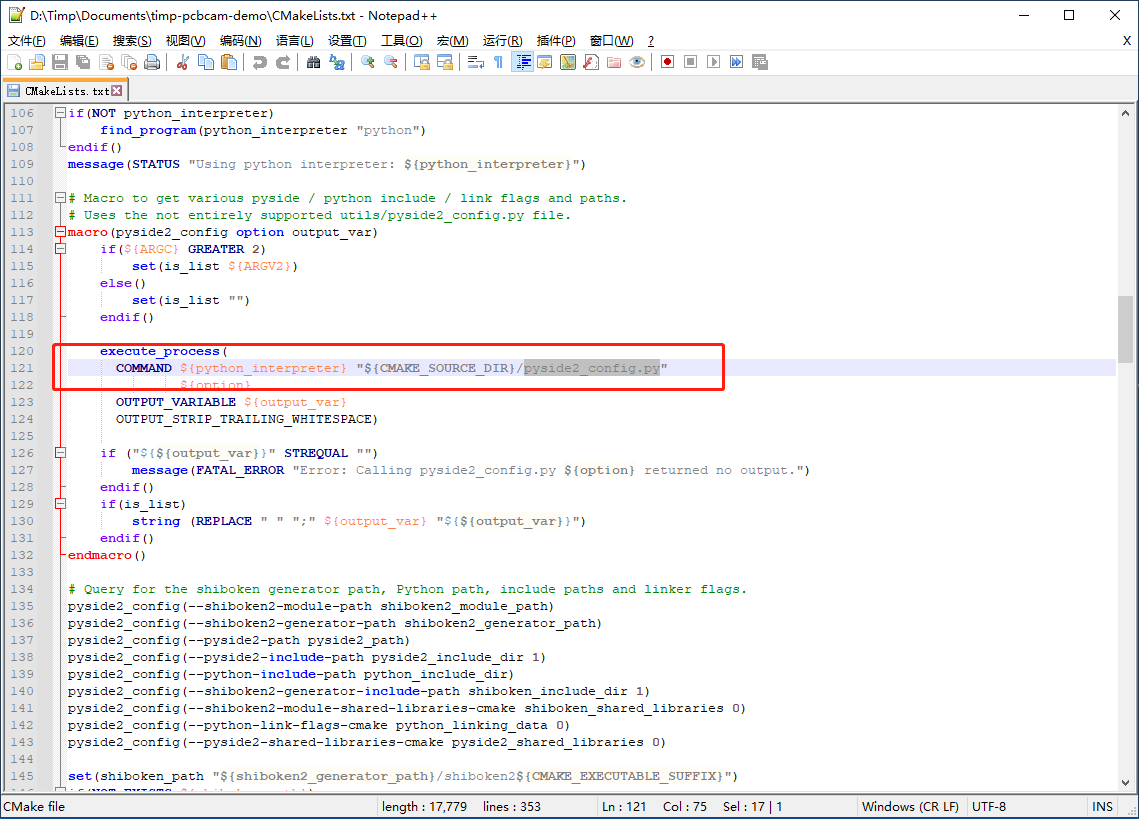
1. 首先指定转化的模块，格式为xx\_module\_wrapper.cpp



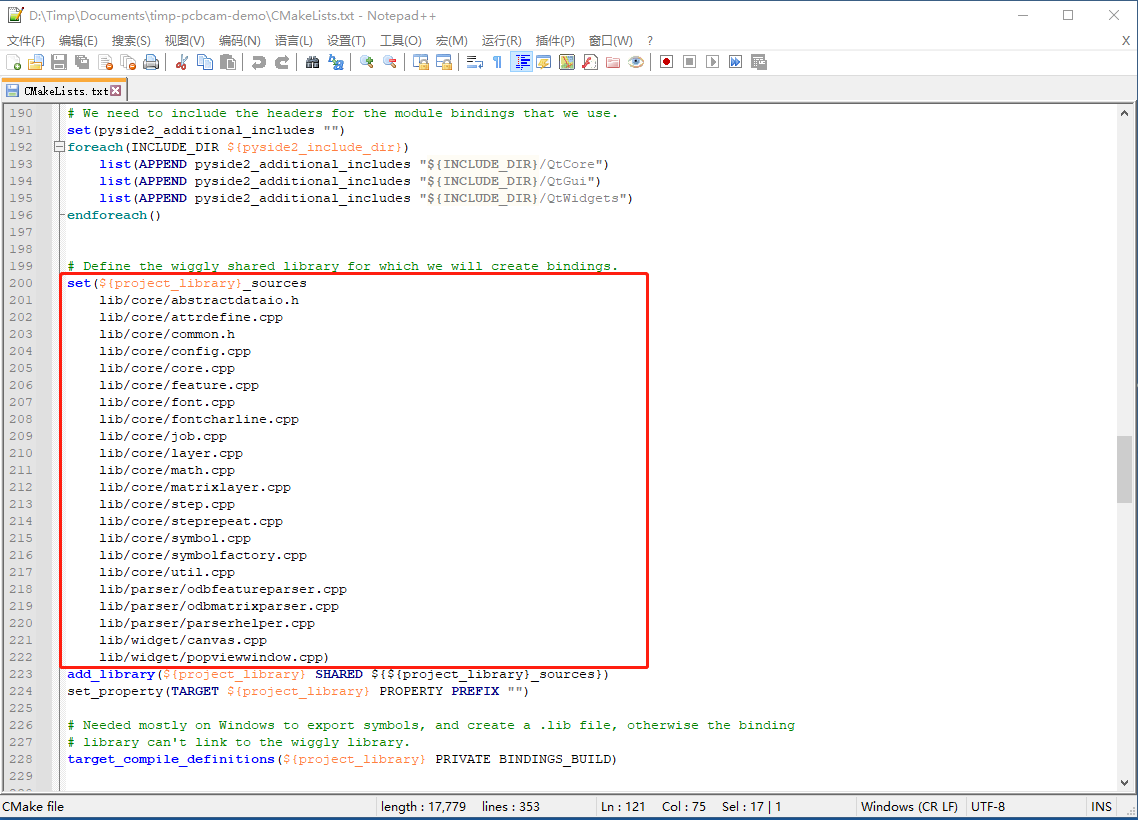
1. 指定需要转换的c++类，格式为xx\_wrapper.cpp，特别需要注意，如果是内部类或者内部结构体，格式为 外部类\_内部类\_wrapper.cpp



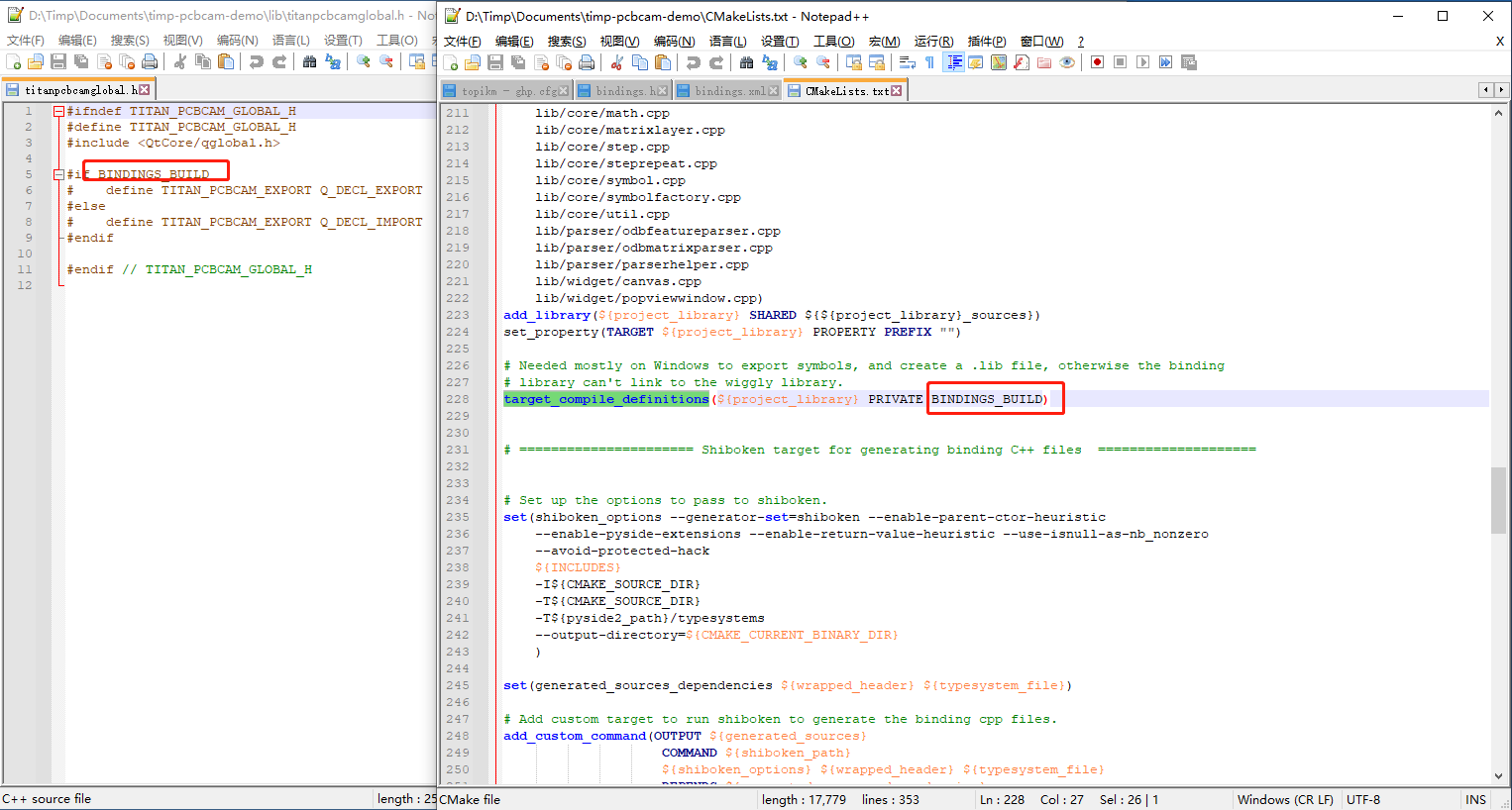
* 修改pyside2\_config.py路径



* 指定源文件

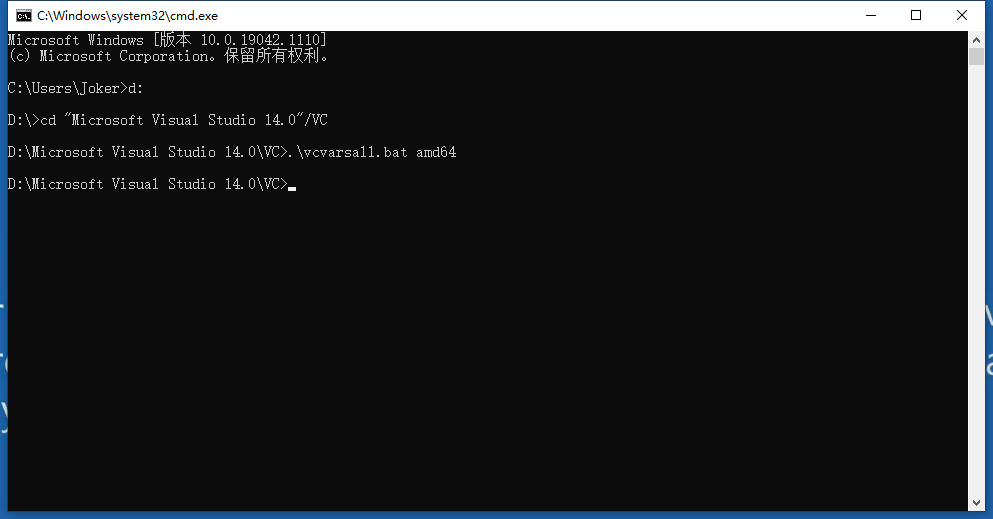


* 修改target\_compile\_definitions，此处需要与c++导出配置名称相同

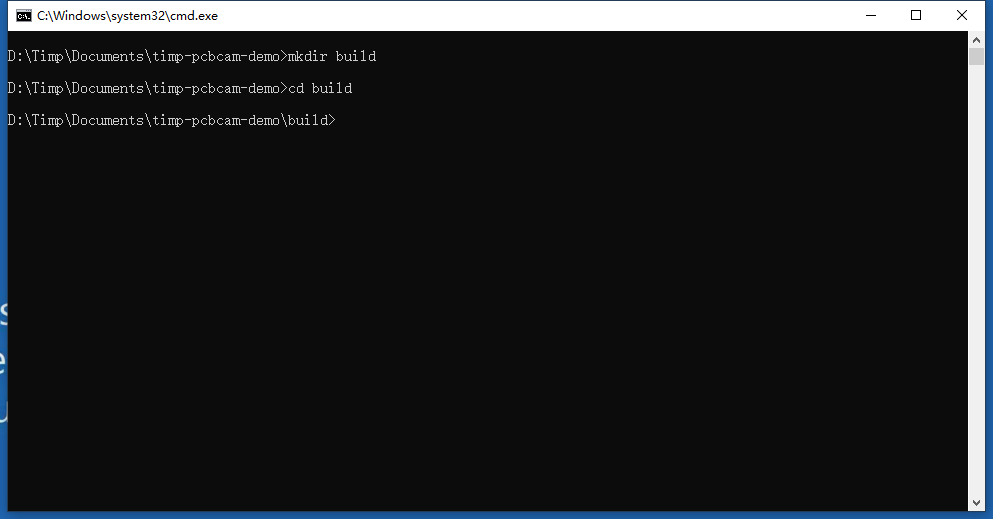


编译

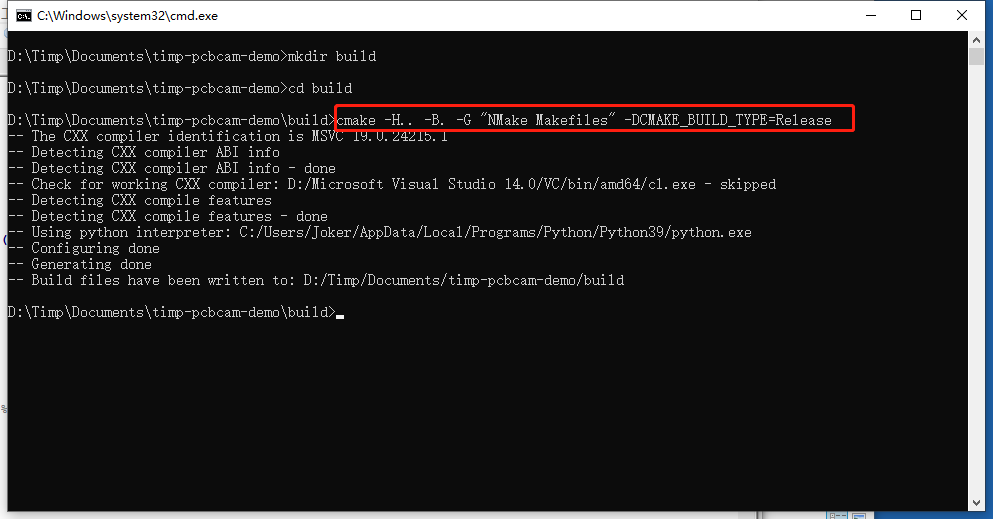
1. 初始化编译环境



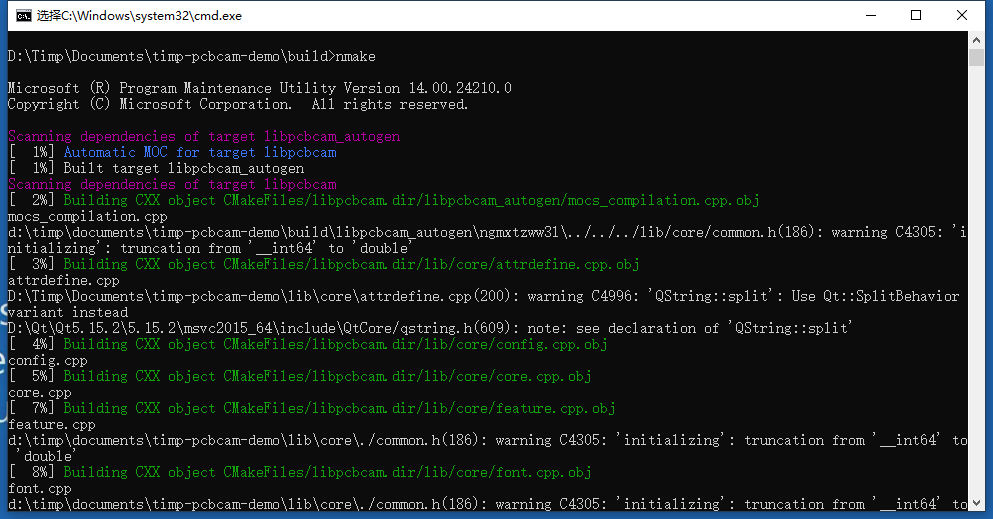
1. 进入项目根目录，创建build目录， 进入build目录

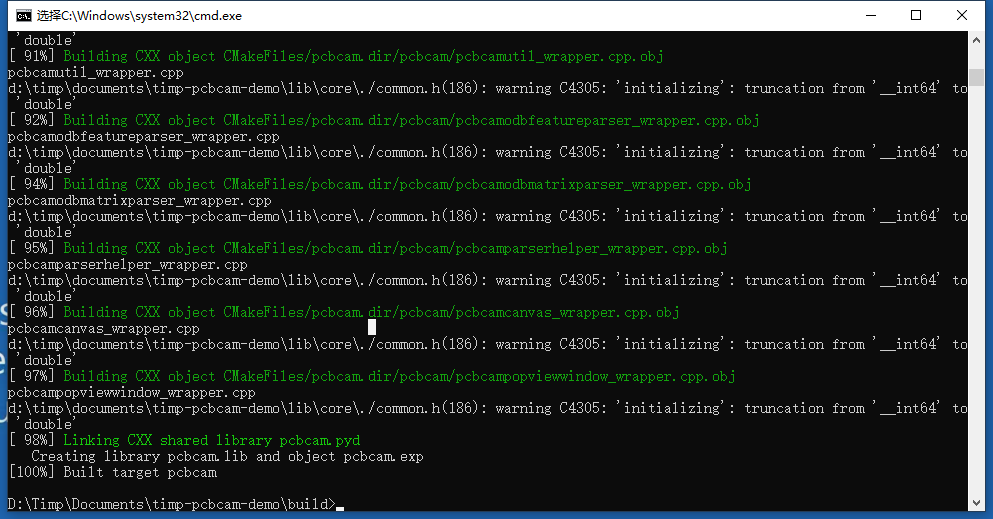


1. 生成nmakefiles，cmake -H.. -B. -G "NMake Makefiles" -DCMAKE\_BUILD\_TYPE=Release



1. 编译， nmake,





1. 编译安装，nmake install。生成的pcbcam.pyd、libpcbcam.dll、shiboken2.abi3.dll即是我们做的扩展库

