# **Windows下使用wsl2编译安卓12系统并调试**

## [WSL2](https://blog.csdn.net/baidu_24392053/article/details/119081623)

运行要求

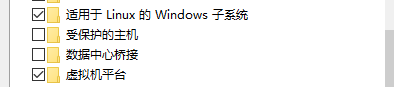
* windows 10
* 对于 x64 系统：版本 1903 或更高版本，采用 内部版本 18362 或更高版本
* 对于 ARM64 系统：**版本 2004** 或更高版本，采用 **内部版本 19041** 或更高版本。
* 低于 18362 的版本不支持 WSL 2。 使用 [Windows Update 助手](https://www.microsoft.com/software-download/windows10)更新 Windows 版本。

若要检查 Windows 版本及内部版本号，选择 Windows 徽标键 + R，然后键入“winver”，选择“确定”。 更新到“设置”菜单中的最新 Windows 版本。



### **启用windows设置**

打开 控制面板->程序和功能->启动和关闭 windows 功能,勾上'适用于linux的Windows 子系统' 和 '虚拟机平台'



### **下载Linux内核更新包**

[适用于x64计算机的WSL2 Linux内核更新包](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi)下载更新包之后运行安装

### **Microsoft Store 中下载Linux 分发版**

下载之前运行命令wsl --set-default-version 2 ,设置安装新的linux发行版时使用wsl2

打开[Microsoft Store](https://aka.ms/wslstore),自行搜索Ubuntu版本下载或者使用[Ubuntu 20.04 LTS](https://www.microsoft.com/store/apps/9n6svws3rx71).

### **WSL2迁移**

wsl2 默认安装在C盘，因为Android 10 系统源码比较庞大，下载加编译需要100多G空间，而且文件系统必须要用linux的，必须在wsl2 unbutu系统内部的文件夹下载编译AOSP，不要下载到window 的文件系统里，比如下载到挂载点 /mnt/d 也就是对应window文件系统的D盘，下载是成功的，但是编译过程会因为跨了window和linux文件系统交叉，会编译出错。系统源码要下载到wsl 虚拟机内部空间里头，所以需要做一个操作：把WSL2迁移到一个空间至少有200G剩余的磁盘里

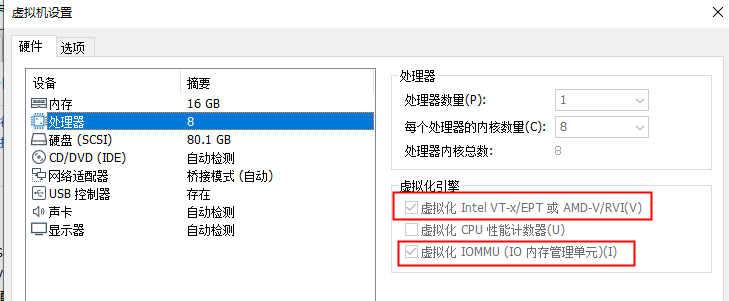
1.查看Linux分发版本Name  
wsl -l --all -v  
NAME STATE VERSION  
Ubuntu-20.04 Stopped 2  
  
2. 导出wsl 内核到非系统分区,例如D盘  
wsl --export Ubuntu-20.04 d:\wsl-ubuntu20.04.tar  
  
3. 注销当前内核  
wsl --unregister Ubuntu-20.04  
  
4. 重新安装 WSL  
wsl --import Ubuntu-20.04 d:\wsl-ubuntu20.04 d:\wsl-ubuntu20.04.tar --version 2  
  
5. 删除中间缓存文件  
del d:\wsl-ubuntu20.04.tar

### **注意事项**

* [Microsoft Store](https://aka.ms/wslstore)打不开
  + 关掉VPN再试一次
* 因为wsl2 启用了windows自己的虚拟机功能，如果你有vmware pro 12等虚拟机开启了将可能会无法启动，wsl 1则不影响, vmware pro 12提示错误：该主机cpu类型不支持虚拟化性能计数器，开启模块VPMC的操作失败，未能启动虚拟机。

解决方法:

去掉下方的勾选

* + 

## **Android系统源码下载**

首先按照后面的博客更新默认镜像源([更新ubuntu 20.04默认镜像源](https://blog.csdn.net/m0_37755995/article/details/128906054)),记得使用清华源

# 切换到主目录下  
cd ~/  
  
# 更新  
sudo apt-get update  
  
#安装git   
sudo apt-get install git   
  
# 下载代码，由于安装好的Ubuntu系统python版本是3.x，repo依赖2.x，所以需要先安装2.x版本的  
sudo apt install python  
  
# 初始化repo环境  
mkdir ~/bin  
  
# 下载repo脚本  
curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo -o repo  
chmod +x repo  
cp repo ~/bin  
  
# 环境变量配置修改  
vim ~/.bashrc  
  
#添加下面两行环境变量配置 !wq 保存退出  
PATH=~/bin:$PATH  
export REPO\_URL='https://mirrors.tuna.tsinghua.edu.cn/git/git-repo'  
  
# 环境变量生效  
source ~/.bashrc  
  
#下载到这个目录aosp下面  
mkdir android-12.1.0\_r11   
cd android-12.1.0\_r11   
# 选择源码版本并下载,这里使用的是android-12.1.0\_r11,从下面的列表网站里面选  
repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest -b android-12.1.0\_r11  
# 开始下载  
repo sync -c  
  
#可选操作： 下载成功后，删除.repo，repo隐藏文件会占据比较大的磁盘空间，如果不需要后续同步代码的话，磁盘比较紧张的老铁最好删了repo隐藏文件，省出点空间给编译用  
rm -rf .repo

[安卓源码版本列表](https://source.android.com/docs/setup/about/build-numbers?hl=zh-cn#source-code-tags-and-builds)

## **驱动下载**

[官方文档](https://source.android.google.cn/setup/build/downloading#obtaining-proprietary-binaries)

由于我同步的代码是android-12.1.0\_r11 分支，驱动程序需要在这里[下载](https://developers.google.cn/android/drivers),以我的 pixel 4 xl为例，下载的代码是android-12.1.0\_r11，TAG对应的驱动BUILD ID我们可以在[这里](https://source.android.google.cn/docs/setup/about/build-numbers)查,



因此我需要下载的驱动是



点击 Link 下载两个文件，然后进行解压到代码根目录，然后执行 sh 脚本释放驱动到合适的位置

./extract-google\_devices-coral.sh  
./extract-qcom-coral.sh

在代码根目录执行，使用 D 来向下翻页，直到最后手动输入 **I ACCEPT ,** 二进制文件及其对应的 makefile 将会安装在源代码树的 vendor/ 层次结构中。

## **Android源码编译**

### **安装依赖**

sudo apt-get install git-core gnupg flex bison build-essential zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 libncurses5 lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev libgl1-mesa-dev libxml2-utils xsltproc unzip fontconfig

### **开编!**

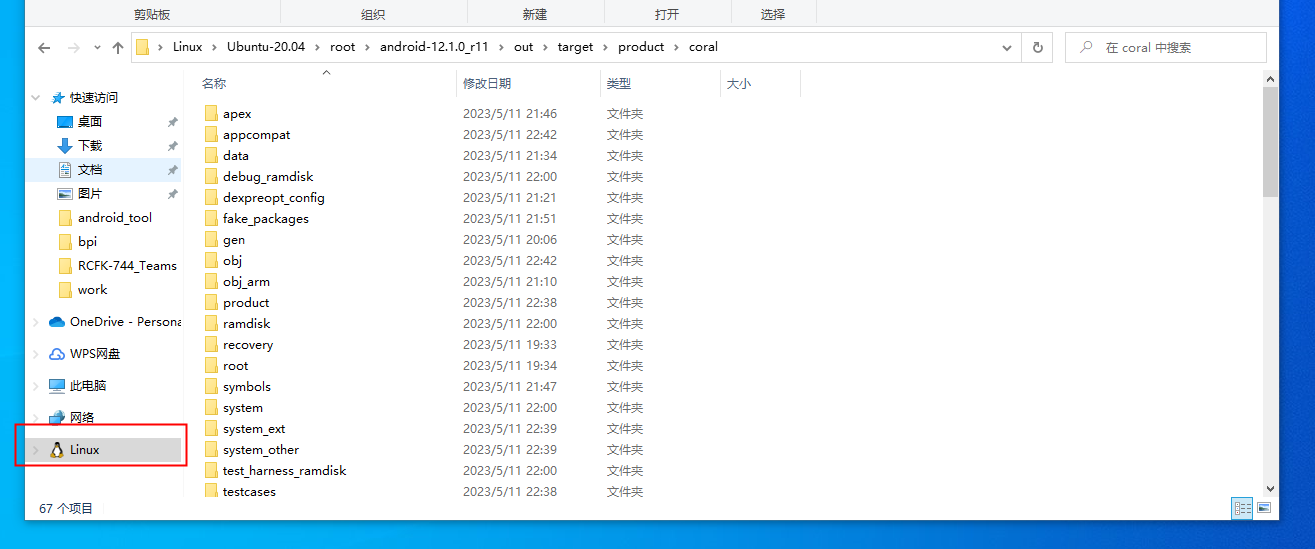
# 在源码根目录执行  
source build/envsetup.sh  
  
# 选择编译目标  
lunch  
{  
You're building on Linux  
  
Lunch menu... pick a combo:  
 ...  
 23. aosp\_coral-userdebug  
 ...  
# 选择自己需要的机器代号,我的是Pixel 4XL,代号是coral,选择23.  
}  
# 编译  
make -j8

### **开刷!**

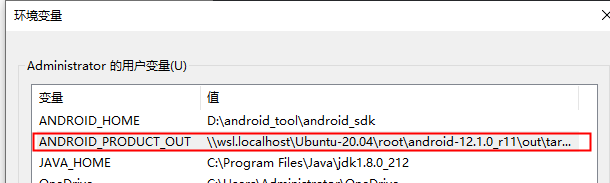
adb reboot fastboot  
  
# 等待手机进入 fastboot 界面之后  
fastboot flashall -w  
  
# 刷机完成之后，执行 fastboot reboot 长期系统即可  
fastboot reboot

注意!

执行到fastboot flashall -w这里时,fastboot死活连不上手机,无法再WSL2中使用fastboot刷入,但是可以在这个文件系统里面找到编译好的输出目录\\wsl.localhost\Ubuntu-20.04\root\android-12.1.0\_r11\out\target\product\coral,shift+右键 从此处打开PowerShell,然后执行命令fastboot flashall -w



执行命令时还是会出错,需要新增一个环境变量ANDROID\_PRODUCT\_OUT变量值为输出目录\\wsl.localhost\Ubuntu-20.04\root\android-12.1.0\_r11\out\target\product\coral



继续执行命令fastboot flashall -w,显示刷入成功

## **Android源码调试**

### **Java层Framwork调试**

###### 编译idegen

idegen是一个 Android 工具，可以生成适用于不同集成开发环境（IDE）的项目文件，例如 Eclipse 和 Android Studio。这个工具的作用是自动生成 IDE 项目文件，以便更方便地在 IDE 中进行开发

# 在源码根目录执行  
source build/envsetup.sh  
  
# id为 aosp\_coral-userdebug ,可以先执行单一lunch指令再选择合适的id  
lunch 23  
  
# 编译指定的目录或模块  
mmm development/tools/idegen  
  
# or make idegen  
sudo development/tools/idegen/idegen.sh  
sudo chmod 777 android.iml   
sudo chmod 777 android.ipr

注意:我是在代码编译完成之后才使用mmm 编译idegen,请尽量在代码全部编译完成后编译idegen

###### 编辑导入配置

android.iml 记录项目所包含的module、依赖关系、SDK版本等等，类似一个XML文件

android.ipr 工程的具体配置，代码以及依赖的lib等信息，类似于Visual Studio的sln文件

在源码根目录执行vim android.iml,输入/excludeFolder搜索到这个位置,再加入下面的配置,用于过滤不需要导入的模块.

<excludeFolder url="file://$MODULE\_DIR$/.repo" />  
 <excludeFolder url="file://$MODULE\_DIR$/art" />  
 <excludeFolder url="file://$MODULE\_DIR$/bionic" />  
 <excludeFolder url="file://$MODULE\_DIR$/bootable" />  
 <excludeFolder url="file://$MODULE\_DIR$/build" />  
 <excludeFolder url="file://$MODULE\_DIR$/compatibility" />  
 <excludeFolder url="file://$MODULE\_DIR$/dalvik" />  
 <excludeFolder url="file://$MODULE\_DIR$/developers" />  
 <excludeFolder url="file://$MODULE\_DIR$/developers/samples" />  
 <excludeFolder url="file://$MODULE\_DIR$/development" />  
 <excludeFolder url="file://$MODULE\_DIR$/device/google" />  
 <excludeFolder url="file://$MODULE\_DIR$/device/sample" />  
 <excludeFolder url="file://$MODULE\_DIR$/docs" />  
 <excludeFolder url="file://$MODULE\_DIR$/external" />  
 <excludeFolder url="file://$MODULE\_DIR$/flashing-files" />  
 <excludeFolder url="file://$MODULE\_DIR$/frameworks/base/docs" />  
 <excludeFolder url="file://$MODULE\_DIR$/kernel" />  
 <excludeFolder url="file://$MODULE\_DIR$/libcore" />  
 <excludeFolder url="file://$MODULE\_DIR$/libnativehelper" />  
 <excludeFolder url="file://$MODULE\_DIR$/out" />  
 <excludeFolder url="file://$MODULE\_DIR$/pdk" />  
 <excludeFolder url="file://$MODULE\_DIR$/platform\_testing" />  
 <excludeFolder url="file://$MODULE\_DIR$/prebuilt" />  
 <excludeFolder url="file://$MODULE\_DIR$/prebuilts" />  
 <excludeFolder url="file://$MODULE\_DIR$/shortcut-fe" />  
 <excludeFolder url="file://$MODULE\_DIR$/test" />  
 <excludeFolder url="file://$MODULE\_DIR$/toolchain" />  
 <excludeFolder url="file://$MODULE\_DIR$/tools" />

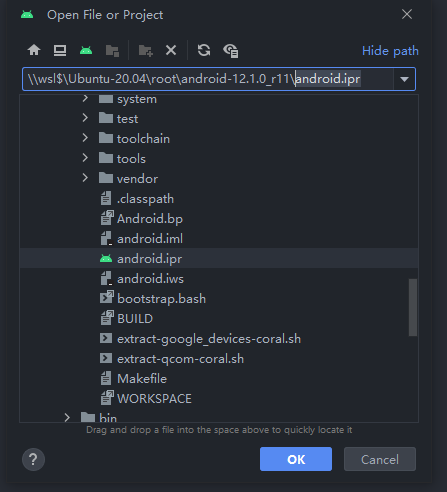
###### 将代码从Ubuntu移出(必须)

将需要调试的frameworks代码和生成的.iml,还有ipr文件一起复制到windows的文件系统下.



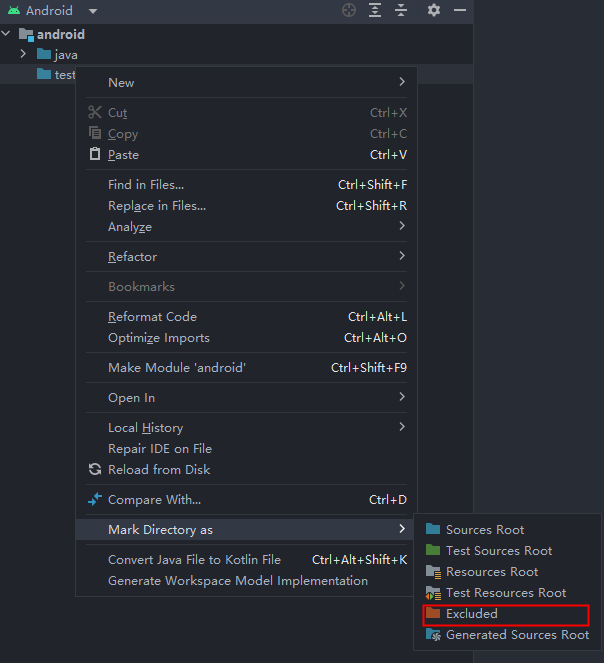
###### 导入Android Studio

点击Android Studio左上角的File -> Open -> 源码目录下的android.ipr,等待加载完成



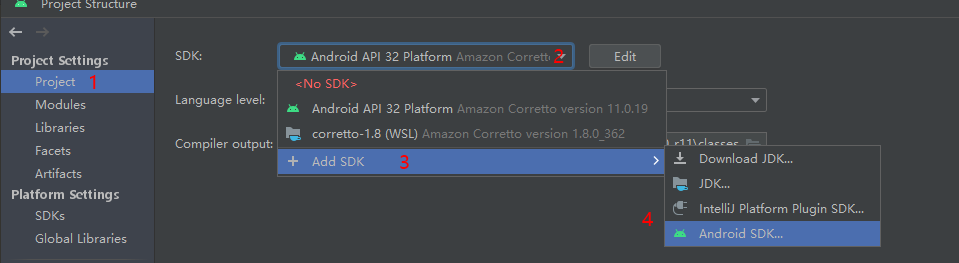
###### 排除tests

tests 目录右键



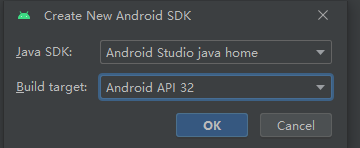
###### 配置 Android源码项目

1. 点击File -> Project Structure–>SDKs配置项目的JDK、SDK。
2. 点击project,点击下拉SDK,选择 Add SDK ,添加一个Android Sdk

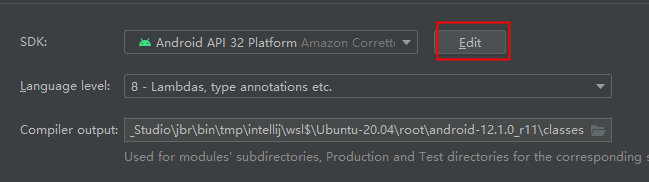


注意:选择SDK必须是Windows里面的SDK

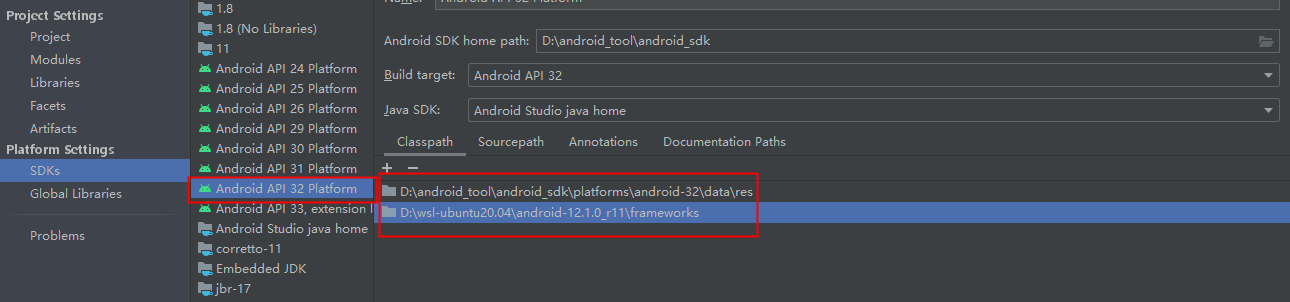
1. JDK选择里面的默认的JDK,Android的版本选择要调试机器的版本API.之后点击Ok

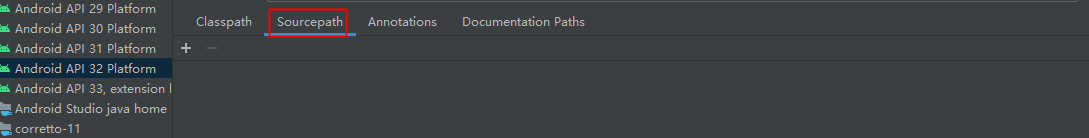


1. 添加完成之后下拉栏就会多一个选项,点击Edit编辑他.

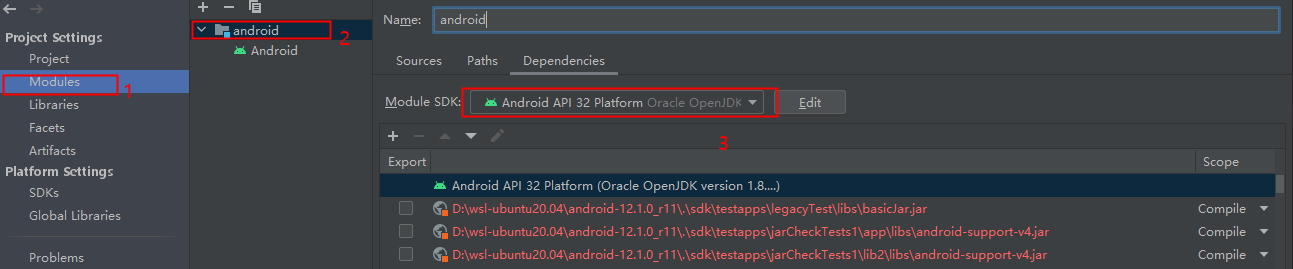


1. 跳到这里之后,选择Classpath,只保留\*\*\data\res选项,点击加号将Framwork代码选择进去,并且点击SourcePath,将里面的路径都清空.

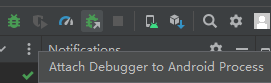


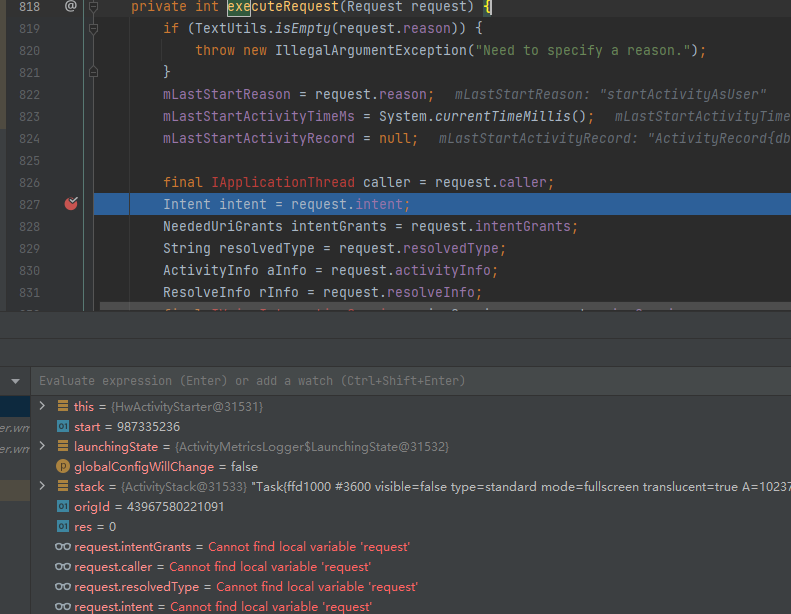


1. 点击Modules,点击android ,下拉选择自己刚刚设置的SDK



###### 开调!

启动应用后,使用附加System\_process,下好executeRequest的断点.可以看到断下来了.



###### 总结

1.虽然文章名字叫wsl上,经过这一次后,不建议在非ubuntu上面搞这些东西,呸麻烦

2.遇到问题要有发散性思维,不能死盯着一篇文章看,要多查查别的文章.

3.我再也不分享这种刷系统类的问题了.