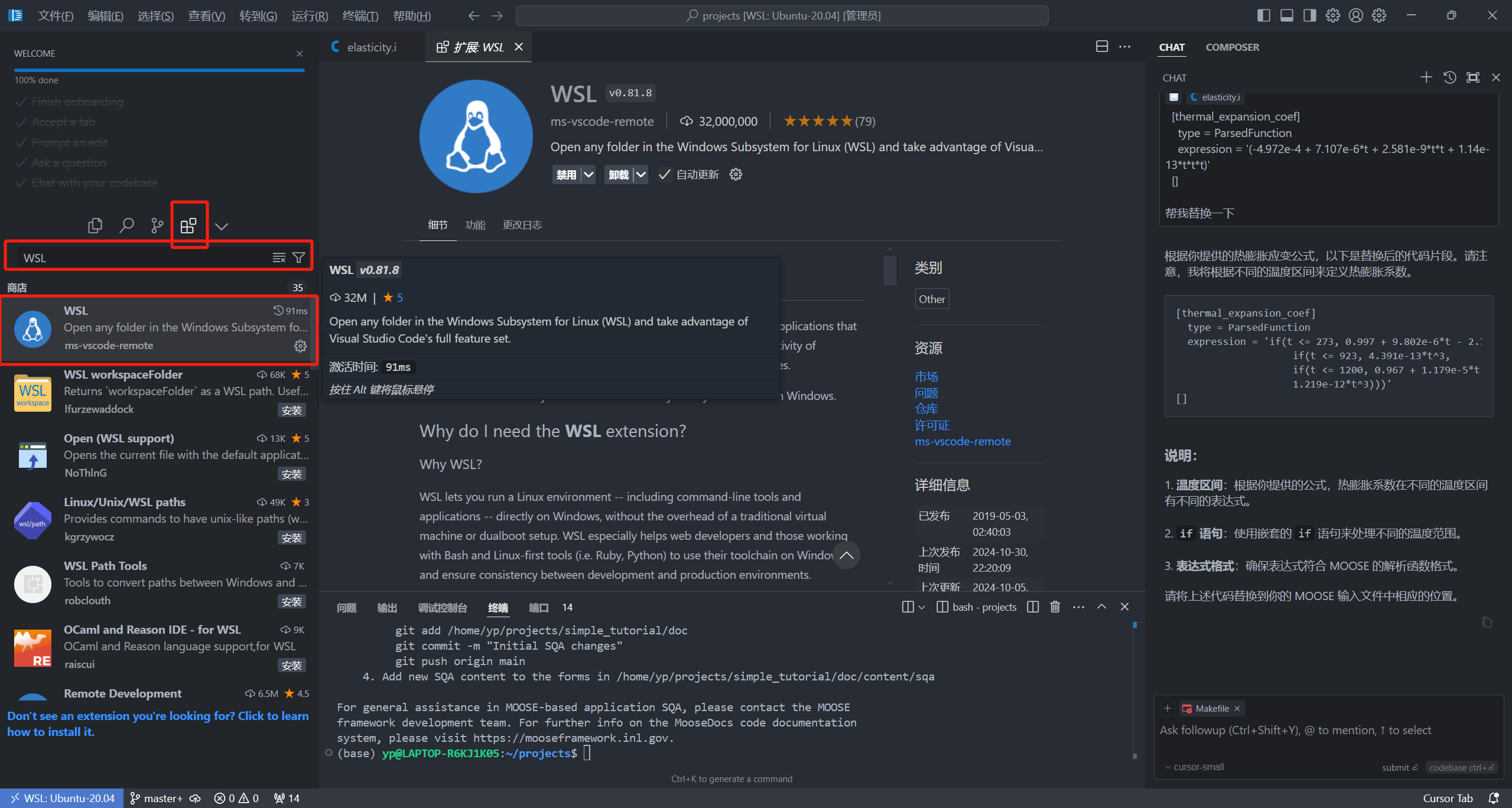
建议先学一下官方教程再回来看我的这个教程，加深印象。

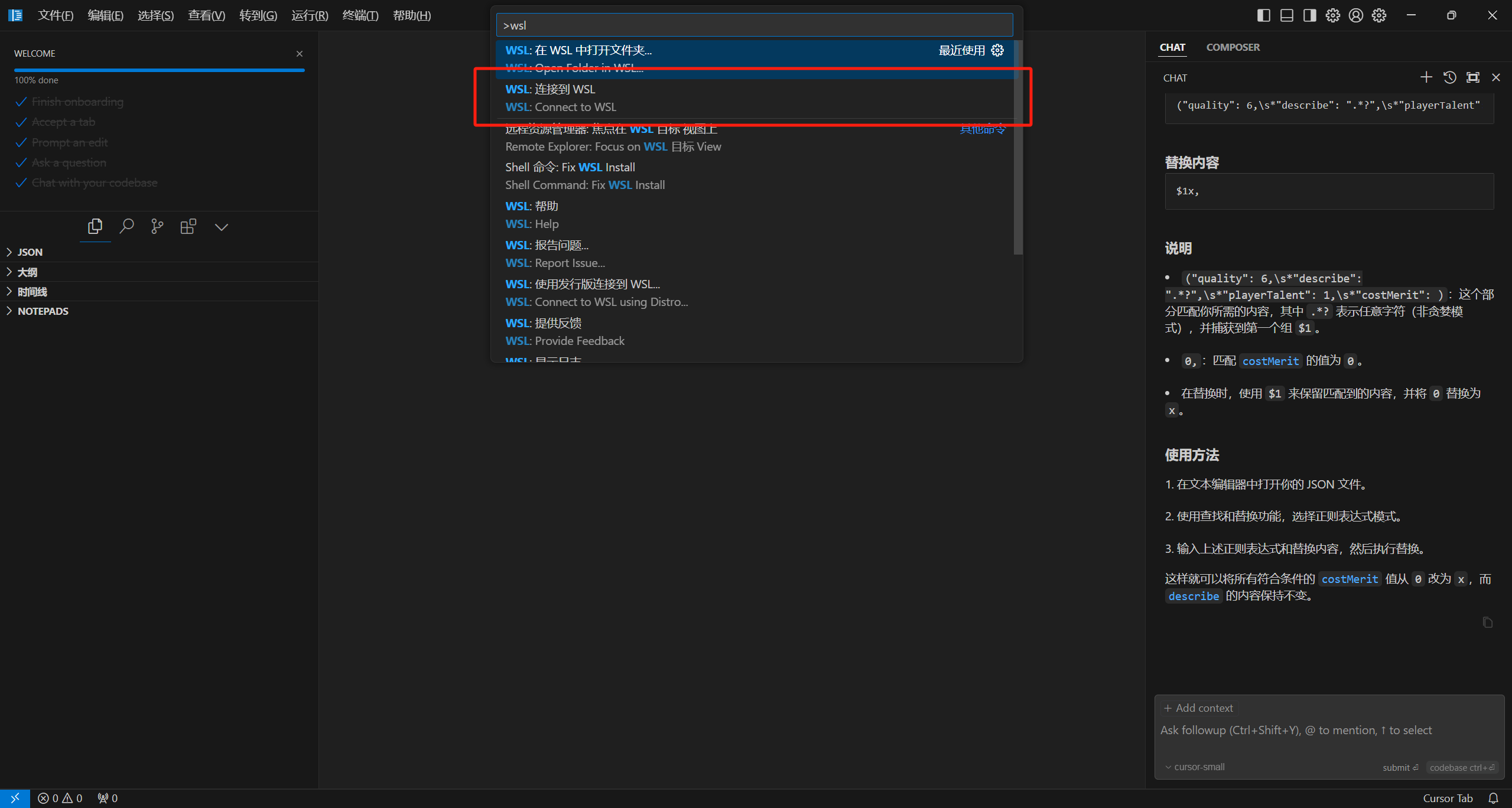
# 安装MOOSE

如果你是Windows系统，需要安装linux子系统，然后安装MOOSE。强烈建议去官网按他的流程来一遍（[Installing MOOSE | MOOSE](https://mooseframework.inl.gov/getting_started/installation/index.html)）。写本教程的时间是2024-12-15，由于版本可能更新，本教程中涉及的代码可能过时而无效或报错。

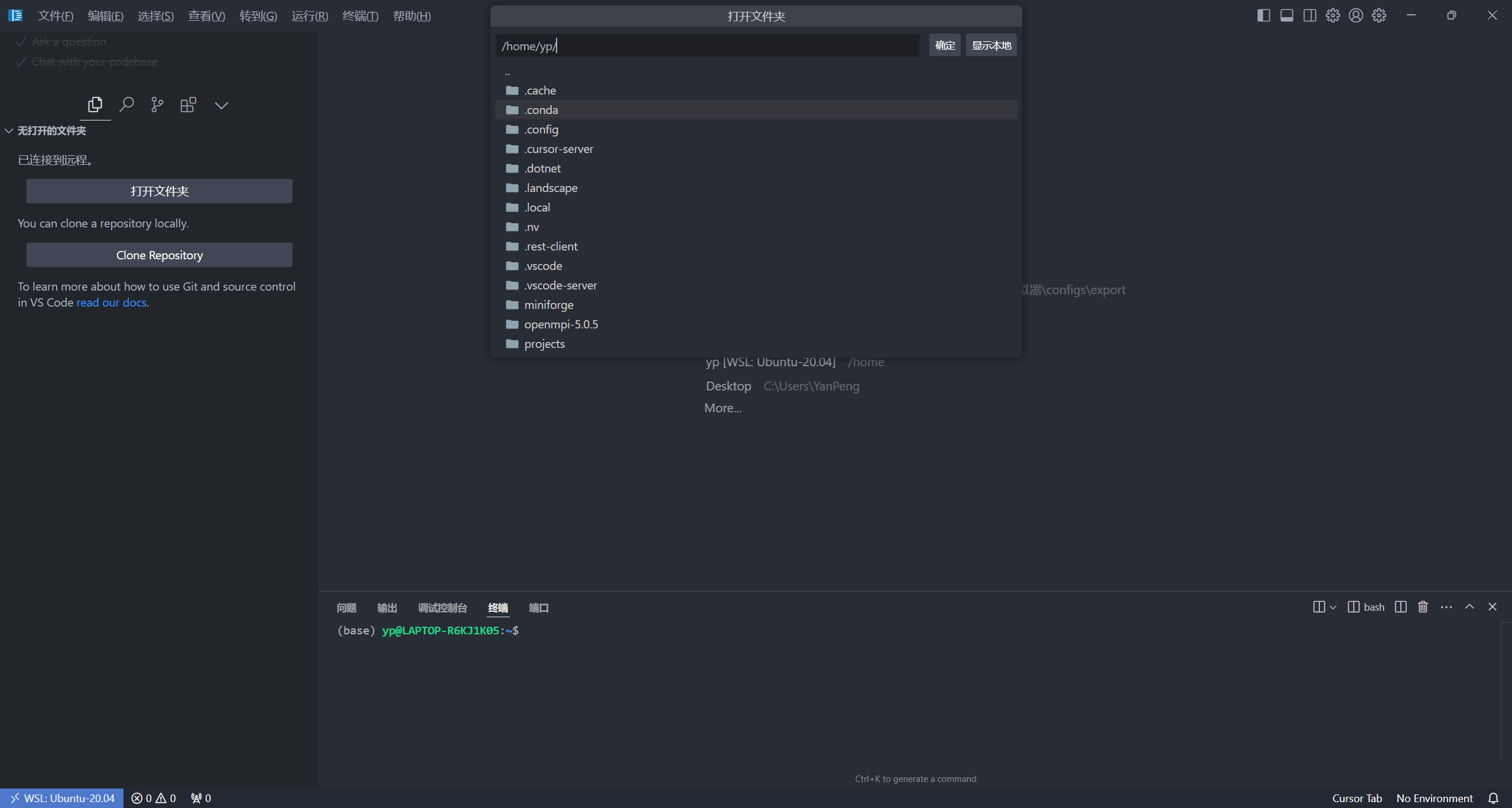
1. 第一步，安装好linux子系统。（[windows11 安装WSL2全流程\_wsl2安装-CSDN博客](https://blog.csdn.net/u011119817/article/details/130745551)，或其他安装WSL2的教程，有一定概率报错，这与每个人的电脑设置有关），按照这个来，安装好linux子系统就OK，可以不用安装图形界面。
   1. [启用window子系统及虚拟化](https://blog.csdn.net/u011119817/article/details/130745551#1window_14)
   2. [下载发行版本](https://aka.ms/wslubuntu2004)
   3. power shell 以管理员方式运行：wsl –update
   4. 双击安装b)中下好的子系统，设置linux系统名字与密码，然后就安好子系统。
2. 第二步，在[Cursor](https://www.cursor.com/)下打开linux子系统。推荐下载[Cursor](https://www.cursor.com/)或者[VsCode](https://code.visualstudio.com/)软件，使用Cursor可直接打开linux子系统的文件夹，方便太多。（这两个软件都集成了ai的代码编辑器功能过于强大，极其推荐，并且他们的节目基本一样，因此下面就介绍Cursor）
   1. 下载[Cursor](https://www.cursor.com/)
   2. 在软件里面安装WSL



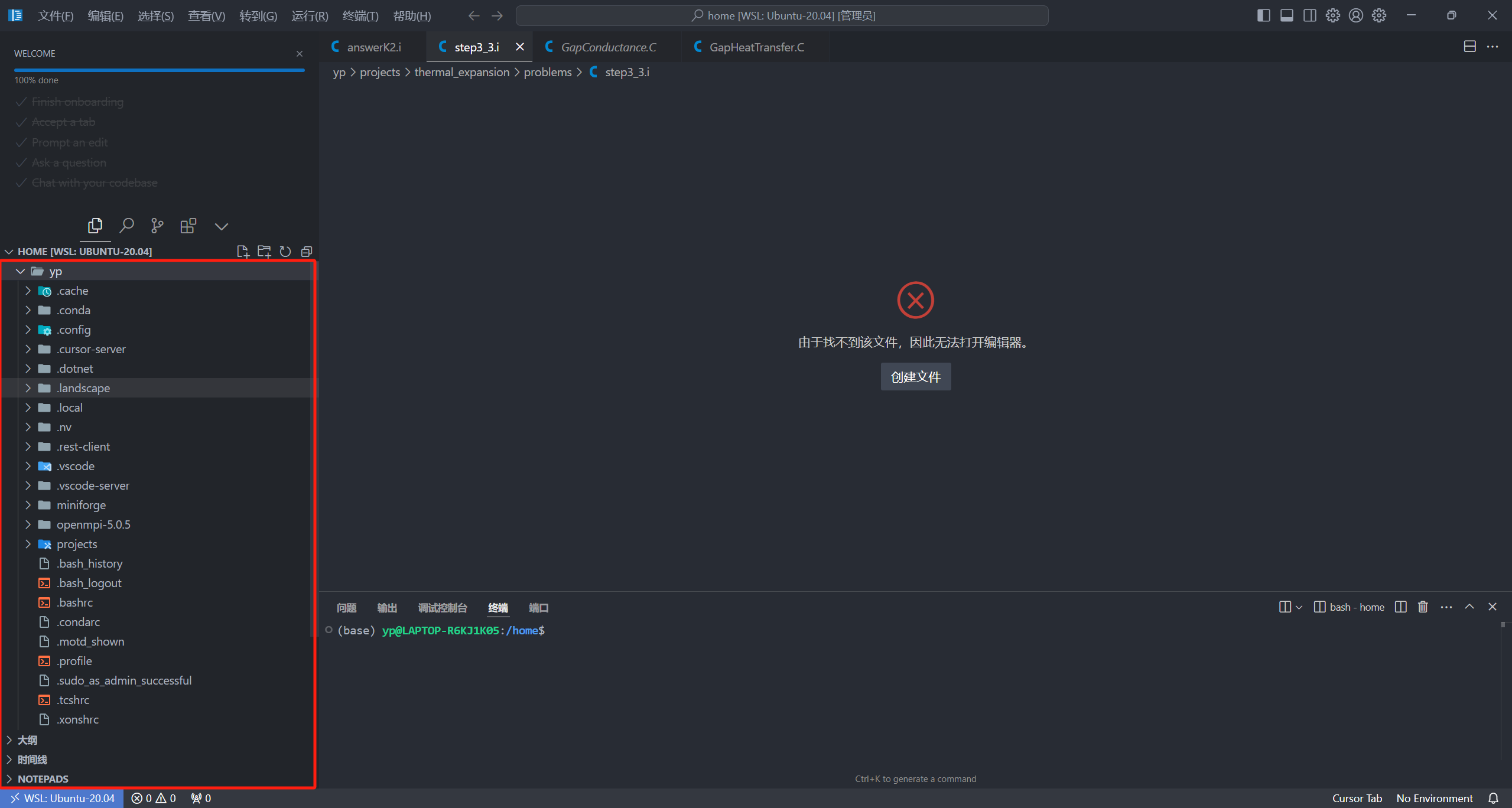
* 1. 输入>WSL，选择连接到WSL



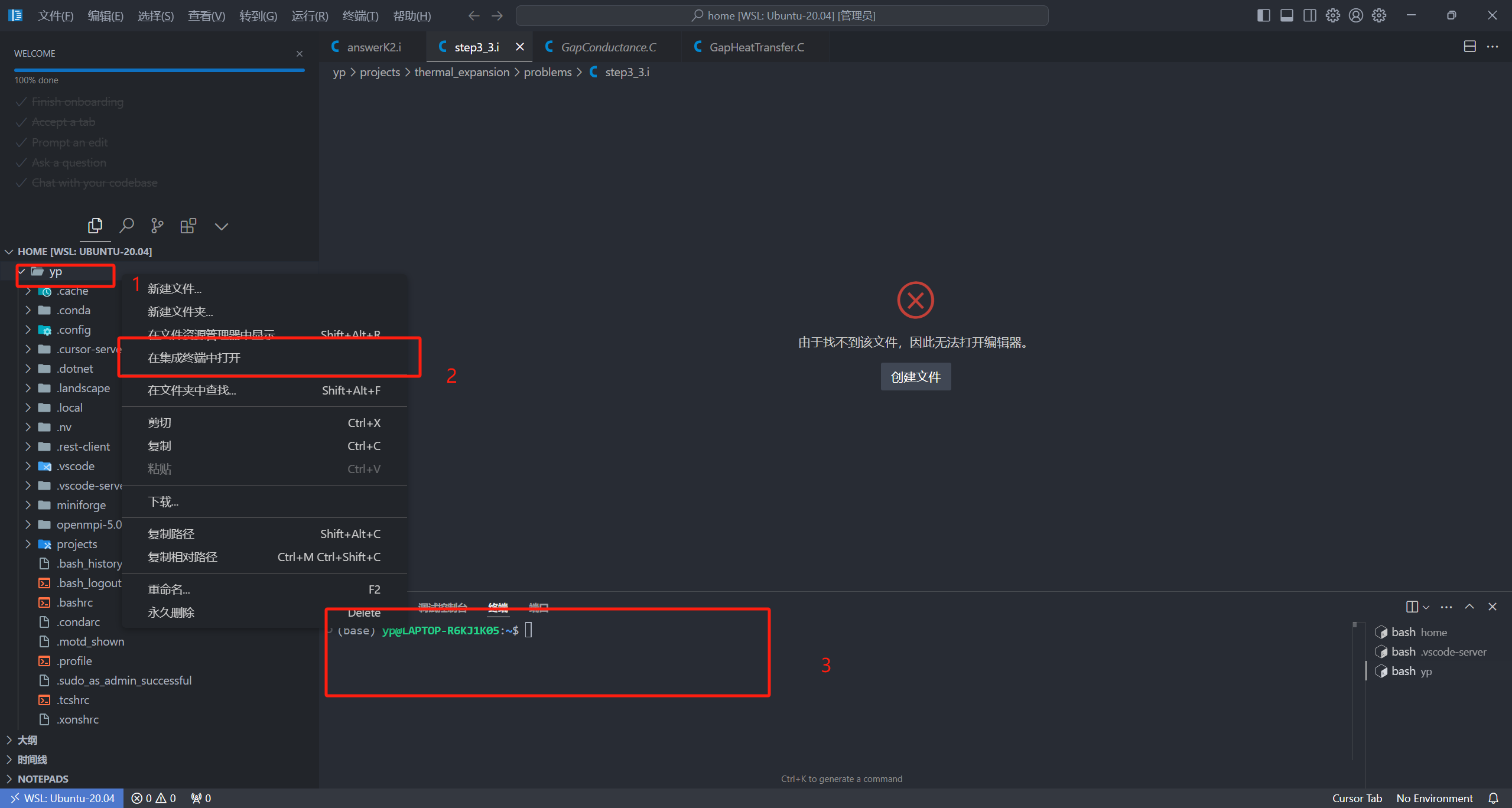
* 1. 然后弹出窗口后，选择打开文件夹/home，点确定



* 1. 出现红色框框。说明你已经配置好了



* 1. 选择《你的linux名字》文件夹，选《在集成终端中打开》，出现3，就说明linux环境配置完成



1. 第三步，配置MOOSE运行环境-python环境安装（[Conda MOOSE Environment | MOOSE](https://mooseframework.inl.gov/getting_started/installation/conda.html)）
   1. 在linux终端处（上图《3》的位置处）中分别输入

《curl -L -O <https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh>》以下载MOOSE环境软件Miniforge3

《bash Miniforge3-Linux-x86\_64.sh -b -p ~/miniforge》以解压MOOSE环境软件miniforge

* 1. 输入《export PATH=$HOME/miniforge/bin:$PATH》以导出环境软件miniforge的路径，目的以后一打开linux终端就能自动启用PATH环境的环境，方便操作
  2. 输入《conda init --all》，你就会发现终端变成了《(base) 名字@XXXX-XXXXXXXX:~$》
  3. 更新python环境《conda update --all --yes》
  4. 输入《conda config --add channels <https://conda.software.inl.gov/public>》以便下载MOOSE运行环境
  5. 输入《conda create -n moose moose-dev=2024.12.02=mpich》安装运行环境（漫长等待），这个环境名字叫做moose，你也可以把它改为其他名字。
  6. 输入《conda activate moose》以激活环境moose（注意现在这个moose只是MOOSE软件的运行环境，现在还没有安装MOOSE软件，每次你要打开MOOSE软件，先输入conda activate moose激活MOOSE的运行环境环境以避免错误）

1. 第四步，安装MOOSE软件
   1. 输入《mkdir -p ~/projects》。在/home/yp路径下创建projects文件夹（自己直接在cursor里右键创建也完全可以）。注意最终是/home/你的linux系统名字/projects
   2. 输入《cd ~/projects》。进入刚才创建的projects文件夹
   3. 输入《git clone https://github.com/idaholab/moose.git》这一步便是抓取MOOSE软件，需要漫长的等待
   4. 输入《cd moose》进入moose软件
   5. 输入《git checkout master》将当前工作目录切换到moose的主分支，（输入就好了，可无视这个解释）（通常称为 master 分支）
2. 第五步，验证MOOSE软件
   1. 输入《cd ~/projects/moose/test》，进入MOOSE软件的test目录
   2. 输入《make -j 6》，编译，漫长的等待，注意，这个《make -j 6》这个蓝色6是根据你的cpu内核数来的，合适的配置可以加快编译与程序运行。编译test文件夹下面的make文件以导出我们最为需要的-opt文件，这个以-opt结尾的文件（例如全称为moose\_test-opt）是我们运行MOOSE输入文件(xxx.i)的必要文件。(可以这样理解，make编译的是我们的整个模型，编译结果是-opt文件，我们用这个模型(-opt)与输入文件(.i)去运行相应MOOSE程序。之后会反复验证与强调这句话)。

[Windows Subsystem for Linux | MOOSE](https://mooseframework.inl.gov/getting_started/installation/windows.html)

# 创建好自己的app

# 简单介绍问题

以该[文件](file:///D:\Postgraduate\Papers\MOOSE\examples\elastoplasticity.i)为例：

这个文件其实要解决的是二氧化铀燃料芯块的热-力耦合方程。

在MOOSE中，必须要明确要解决的控制方程，如，我这里需要解的是如下两个控制方程：

热传导方程： 1‑1

力平衡方程： 1‑2

接下来，推导得出它的弱形式。（详细请看[Step 4 Generate a Weak Form | MOOSE (inl.gov)](https://mooseframework.inl.gov/getting_started/examples_and_tutorials/tutorial01_app_development/step04_weak_form.html)的推导）。【或可以先不管，主要看右边的项】

如热传导方程：

可以写成MOOSE使用的弱形式： =0

需要知道的是，MOOSE所有弱形式方程中的每一项（，热平衡方程中出现了3项）其实都对应好[Kernels]中的一项与 [Materials]中的若干项（如中的*ADHeatConductionTimeDerivative方法对应，而*这两个材料属性则是与 [Materials]中的若干项（也可能没有，看每一个项中有没有材料参数），）具体而言，对应的[Kernels]与如图 1.1所示。

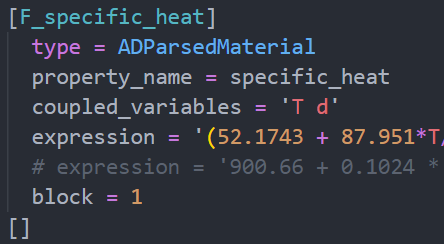
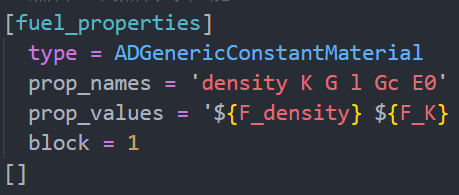
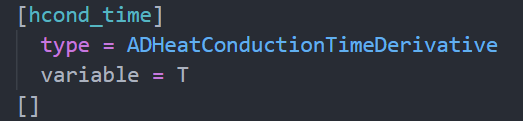


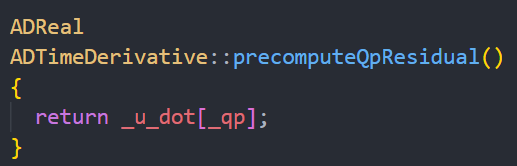
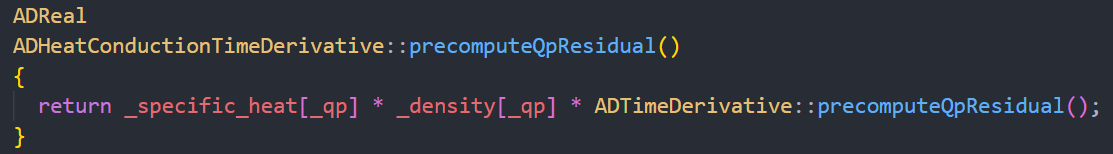
图 1.1 [Kernels]与[Materials]的部分值（左1是[Kernels]的项，右边两个是[Materials]的项）

这里我们再理一下：

1. 所有控制方程
2. 控制方程的弱形式
3. 弱形式的每一项都对应[Kernels]与[Materials]的项

### [Kernels]：

我们找到/moose/modules/heat\_transfer/src/kernels/ADHeatConductionTimeDerivative.C与其调用的ADTimeDerivative.C文件，可以看到计算残差的部分其实与对应的很好。



由于不同的控制方程中有许许多多不同的项，每一个项的形式不一样，为了方便定义与使用，MOOSE里面有非常多[Kernels]以便使用，详细请看[Kernels System | MOOSE](https://mooseframework.inl.gov/moose/syntax/Kernels/)，

本例子中， ADHeatConduction、ADHeatConductionTimeDerivative、HeatSource分别对应

=0

### 一般参数的定义与引用

如同python或者C++，需要命名全局变量或者局部变量一样。在moose输入文件中，我们先将一些材料参数（如热导率，密度等）进行全局变量的定义（此时moose在计算时是不会调用的，需要用特殊的引用格式才能被moose使用）。

首先在此使用全局变量与表达式的进行命名，写在MOOSE的输入文件(.i文件的开头)，例如

density=3980.0*#*kg⋅m-3

thermal\_conductivity=31*#*W⋅m-1⋅K-1

*#* thermal\_conductivity\_dem=0.026#孔隙的导热系数W⋅m-1⋅K-1

elastic\_constants=3.7e11*#*Pa

nu = 0.3

K = '${fparse elastic\_constants/3/(1-2\*nu)}'

G = '${fparse elastic\_constants/2/(1+nu)}'

注意引用的格式一般用 ？ **= '${ elastic\_constants } ${ K}'**

### [Materials]：

其实就是将\_specific\_heat[\_qp] 、\_density[\_qp]等[Kernels]需要调用的参数提前先定义好，可以是常数(ADGenericConstantMaterial方法),可以是与变量耦合的公式（如热导率与温度耦合的经验公式ADParsedMaterial方法），甚至与[Materials]本身模块耦合的公式（ADDerivativeParsedMaterial方法）

下面是[Materials]里面常用的模块与例子：

[fuel\_properties]

type = ADGenericConstantMaterial

prop\_names = 'density K'

prop\_values = '${F\_density} ${F\_K}'

block = 1

[]

在 ADGenericConstantMaterial中，使用

prop\_names = 'x1 x2 x3 x4'与

prop\_values = '${y1} ${y2} y3 y4'

以确定材料参数 x1,x2,x3等值。其中${y1}表示y1是定义的全局变量（即在输入文件开头的density=3980.0*#*kg⋅m-3、nu = 0.3、K = '${fparse elastic\_constants/3/(1-2\*nu)}'、G = '${fparse elastic\_constants/2/(1+nu)}'等

）。

[base\_thermal\_conductivity]

type = ADParsedMaterial

property\_name = base\_conductivity

coupled\_variables = 'T'  *#* T in Kelvin

*#* 使用Harding-Martin表达式

block = 1

expression = '100/(7.5408 + 17.692\*T/1000 + 3.6142\*(T/1000)^2) + 6400/((T/1000)^2.5)\*exp(-16.35/(T/1000))'

[]

在ADParsedMaterial中，使用property\_name确定材料参数的名字。耦合变量为coupled\_variables，公式使用expression。（expression的乘法为\*，次方为^）

*#*热导率

[thermal\_conductivity]

type = ADDerivativeParsedMaterial

property\_name = thermal\_conductivity

coupled\_variables = 'd'

material\_property\_names = 'base\_conductivity'

expression = '(1-d)\*base\_conductivity + d\*base\_conductivity\*0.1'

block = 1

[]

在ADDerivativeParsedMaterial中，也和ADParsedMaterial类似，只是它多了与已经定义好的其他材料参数耦合的量material\_property\_names。