

视觉算法组考核

23211121-陈冠宇

P1Git

P2Linux

1. task1命令行

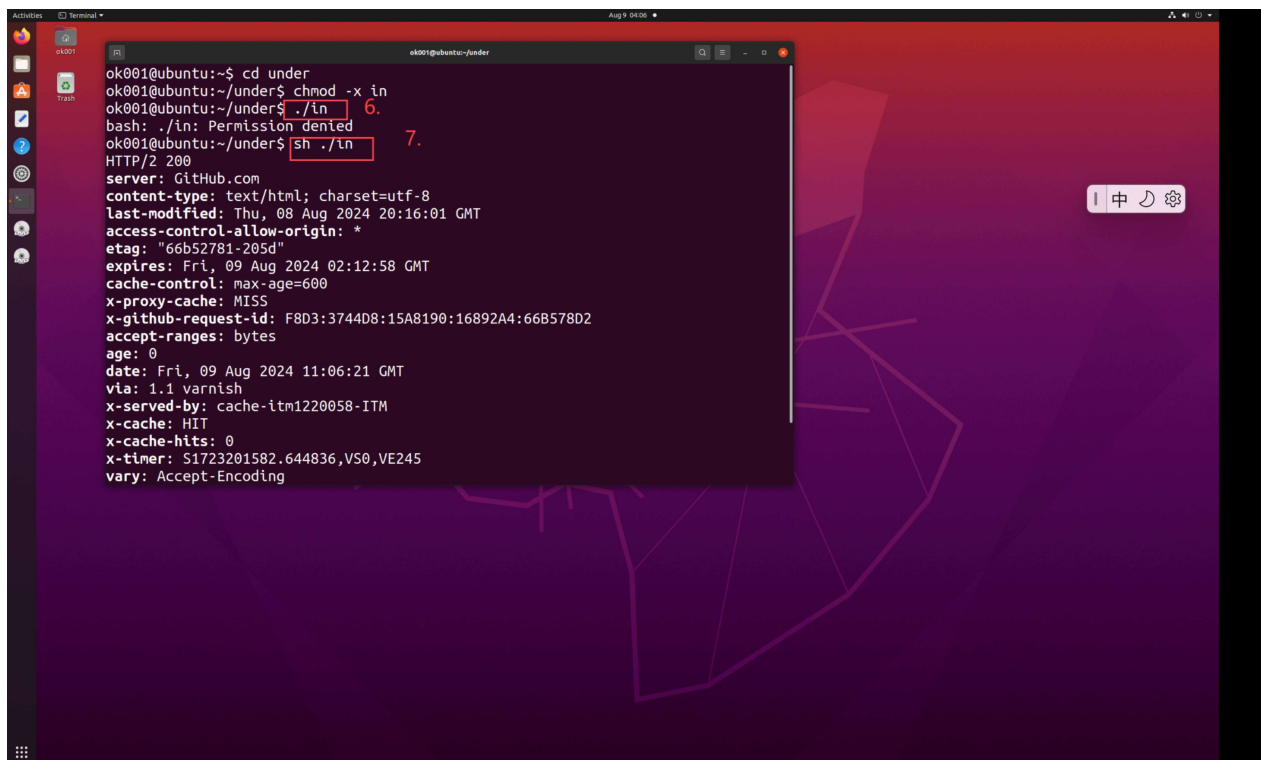
- i. echo返回后面内容的返回值，\$SHELL是个变量名，内容为SHELL编译器的地址
- ii. mkdir产生文件夹under
- iii.
- iv. touch产生文件in
- v. echo回环，>是覆盖输入，>>是追加，且自带换行

```
ok001@ubuntu:~$ rm -r touch
ok001@ubuntu:~$ rm -r missing
ok001@ubuntu:~$ echo $SHELL 1.
/bin/bash
ok001@ubuntu:~$ mkdir under
mkdir: cannot create directory 'under': File exists
ok001@ubuntu:~$ rm -r under
ok001@ubuntu:~$ mkdir under 2.
ok001@ubuntu:~$ cd under
ok001@ubuntu:~/under$ touch in 4.
ok001@ubuntu:~/under$ echo '#!bin/sh' > in
ok001@ubuntu:~/under$ echo 'curl --head --silent https://csail.mit.edu' >> in
ok001@ubuntu:~/under$
```

5.有打错，后来更正了

- vi. ./直接执行，执行失败，反馈不允许

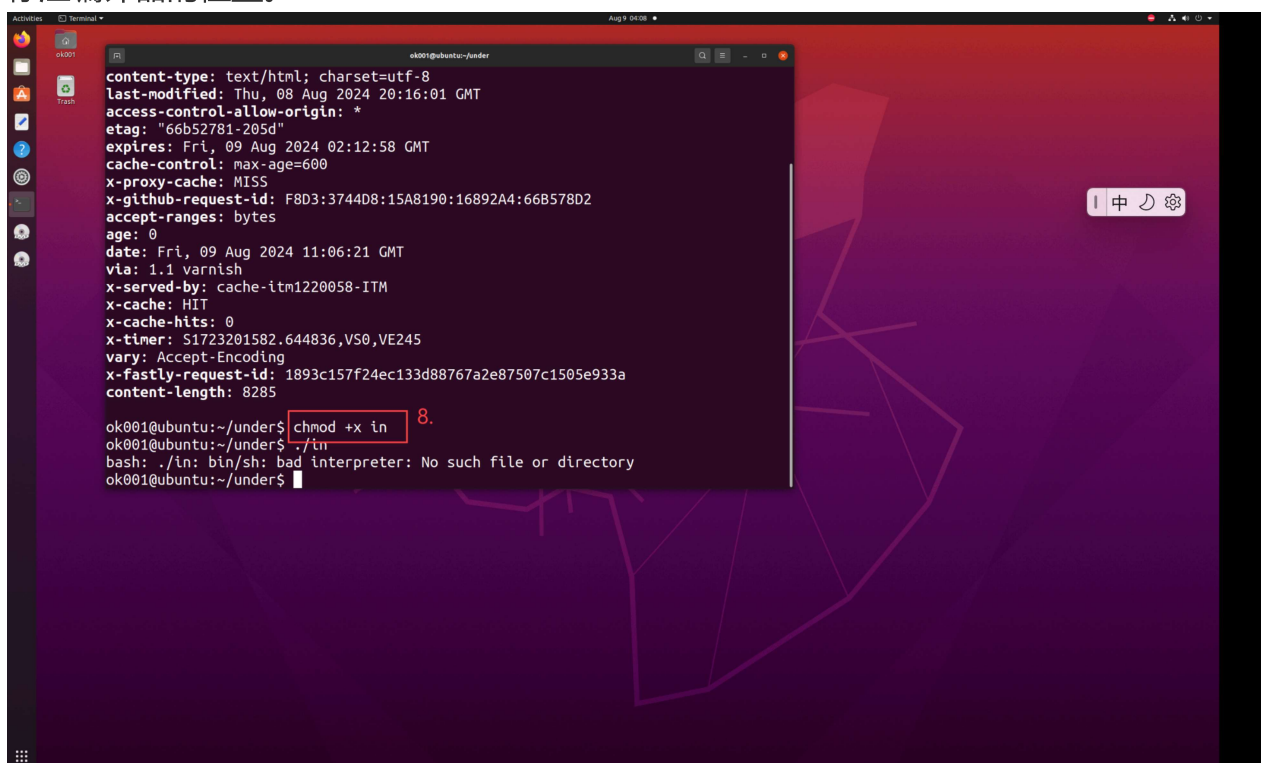
vii. 用sh ./执行成功

A terminal window on a Linux desktop. The user is in the directory ~/under. They run 'cd under' and then 'chmod -x in'. Then they run './in' which fails with 'Permission denied'. Finally, they run 'sh ./in' which succeeds, returning an HTTP 200 status and a list of headers from GitHub.com. Red boxes highlight the commands './in' and 'sh ./in', with red numbers 6 and 7 next to them respectively.

```
ok001@ubuntu:~$ cd under
ok001@ubuntu:~/under$ chmod -x in
ok001@ubuntu:~/under$ ./in 6.
bash: ./in: Permission denied
ok001@ubuntu:~/under$ sh ./in 7.
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8
last-modified: Thu, 08 Aug 2024 20:16:01 GMT
access-control-allow-origin: *
etag: "66b52781-205d"
expires: Fri, 09 Aug 2024 02:12:58 GMT
cache-control: max-age=600
x-proxy-cache: MISS
x-github-request-id: F8D3:3744D8:15A8190:16892A4:66B578D2
accept-ranges: bytes
age: 0
date: Fri, 09 Aug 2024 11:06:21 GMT
via: 1.1 varnish
x-served-by: cache-itm1220058-ITM
x-cache: HIT
x-cache-hits: 0
x-timer: S1723201582.644836,VS0,VE245
vary: Accept-Encoding
```

viii. chmod +x增加文件执行权限后可执行。

ix. 用以上方式就不需要键入了。同时，./就可以运行而不用sh显示运行是因为在执行文件首行有标注编译器的位置。

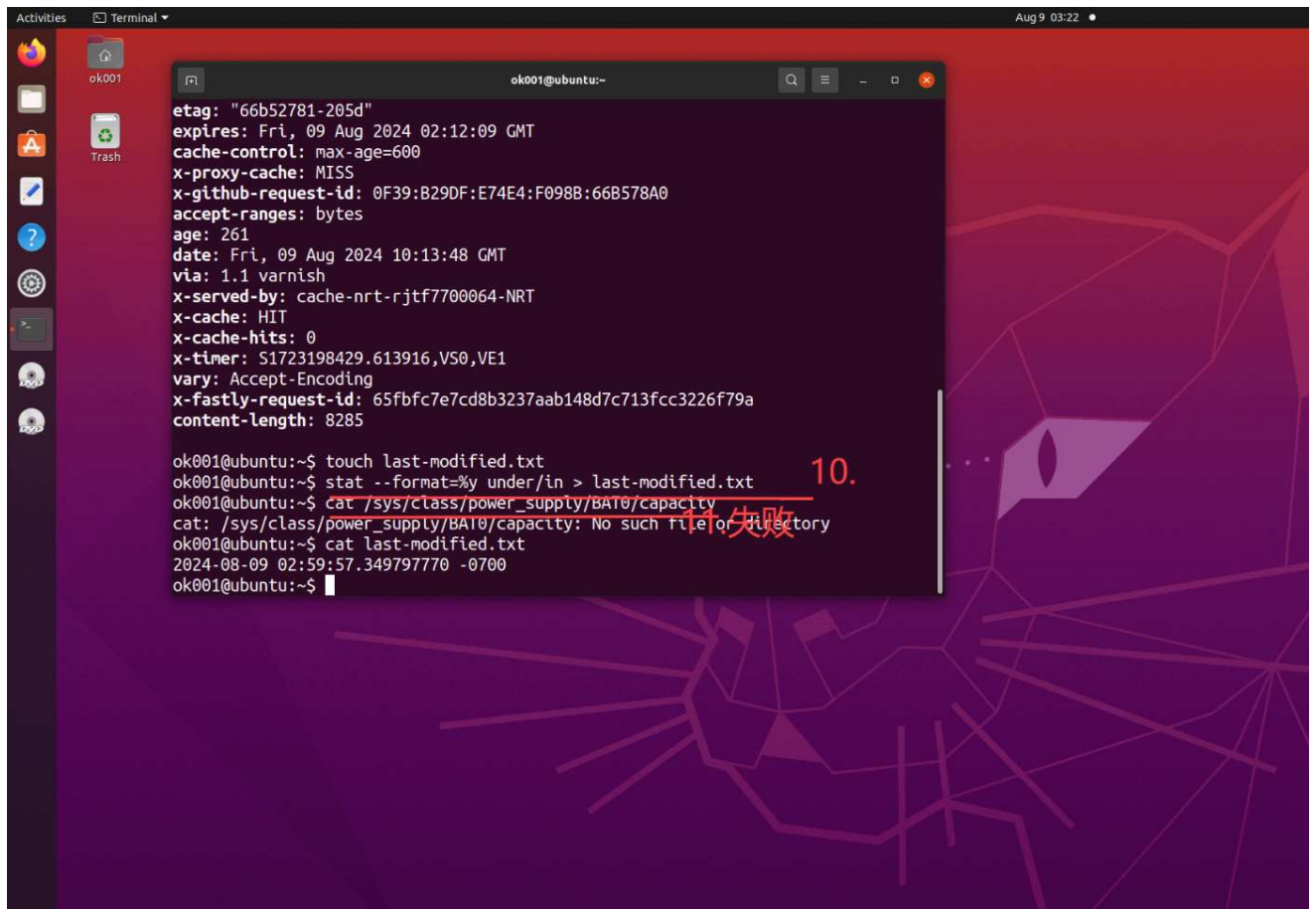
A terminal window showing the same headers as the previous screenshot. Then the user runs 'chmod +x in' and then './in'. The './in' command fails with the error 'bash: ./in: bin/sh: bad interpreter: No such file or directory'. A red box highlights the 'chmod +x in' command with a red number 8 next to it.

```
content-type: text/html; charset=utf-8
last-modified: Thu, 08 Aug 2024 20:16:01 GMT
access-control-allow-origin: *
etag: "66b52781-205d"
expires: Fri, 09 Aug 2024 02:12:58 GMT
cache-control: max-age=600
x-proxy-cache: MISS
x-github-request-id: F8D3:3744D8:15A8190:16892A4:66B578D2
accept-ranges: bytes
age: 0
date: Fri, 09 Aug 2024 11:06:21 GMT
via: 1.1 varnish
x-served-by: cache-itm1220058-ITM
x-cache: HIT
x-cache-hits: 0
x-timer: S1723201582.644836,VS0,VE245
vary: Accept-Encoding
x-fastly-request-id: 1893c157f24ec133d88767a2e87507c1505e933a
content-length: 8285

ok001@ubuntu:~/under$ chmod +x in 8.
ok001@ubuntu:~/under$ ./in
bash: ./in: bin/sh: bad interpreter: No such file or directory
ok001@ubuntu:~/under$
```

x. 显示最后一次编辑的日期

xi. 显示温度，没成功



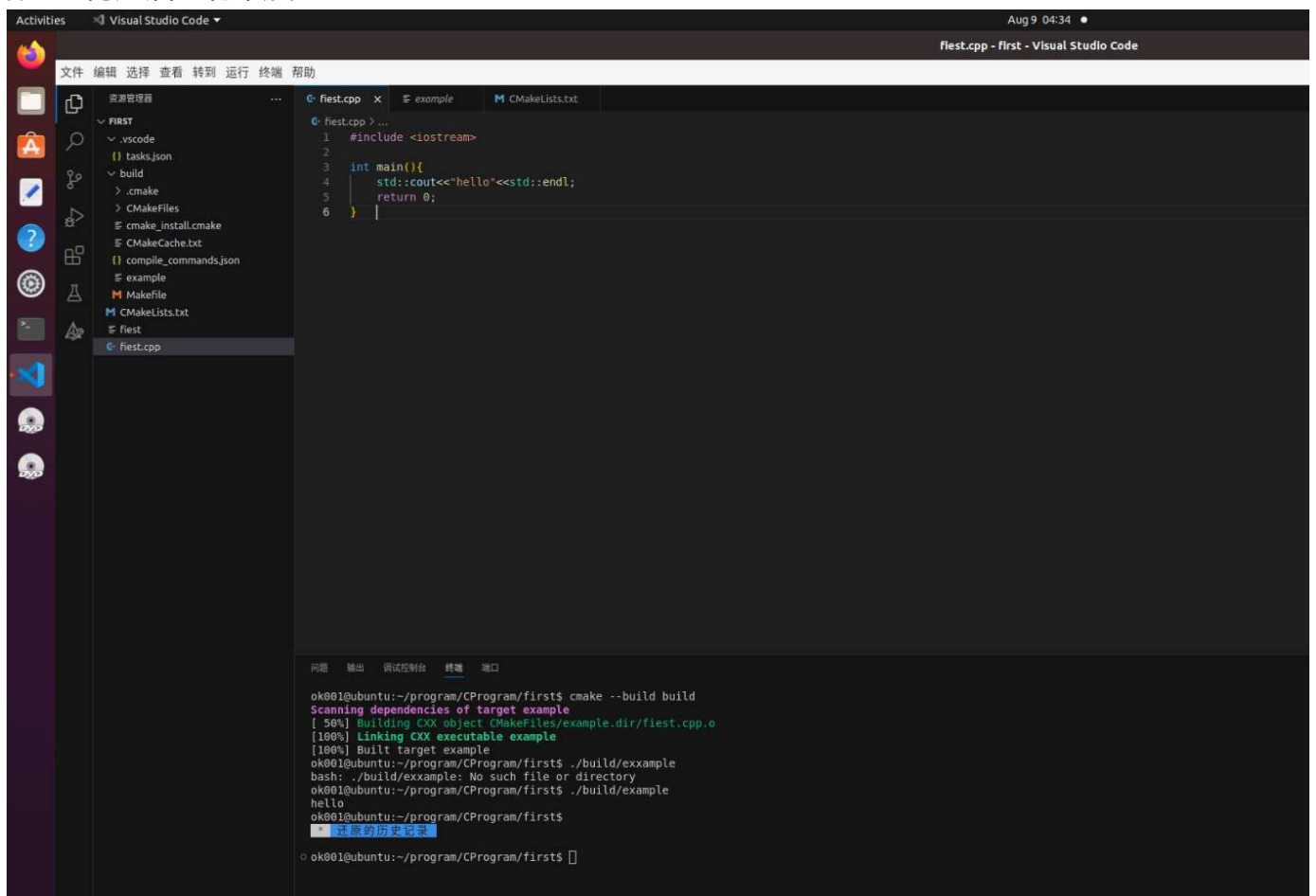
```
ok001@ubuntu:~$ touch last-modified.txt
ok001@ubuntu:~$ stat --format=%y under/in > last-modified.txt
ok001@ubuntu:~$ cat /sys/class/power_supply/BAT0/capacity
cat: /sys/class/power_supply/BAT0/capacity: No such file or directory
ok001@ubuntu:~$ cat last-modified.txt
2024-08-09 02:57:57.349797770 -0700
ok001@ubuntu:~$
```

10. 失败

2. task2markdown

3. task3cmake

配置+构建后运行项目

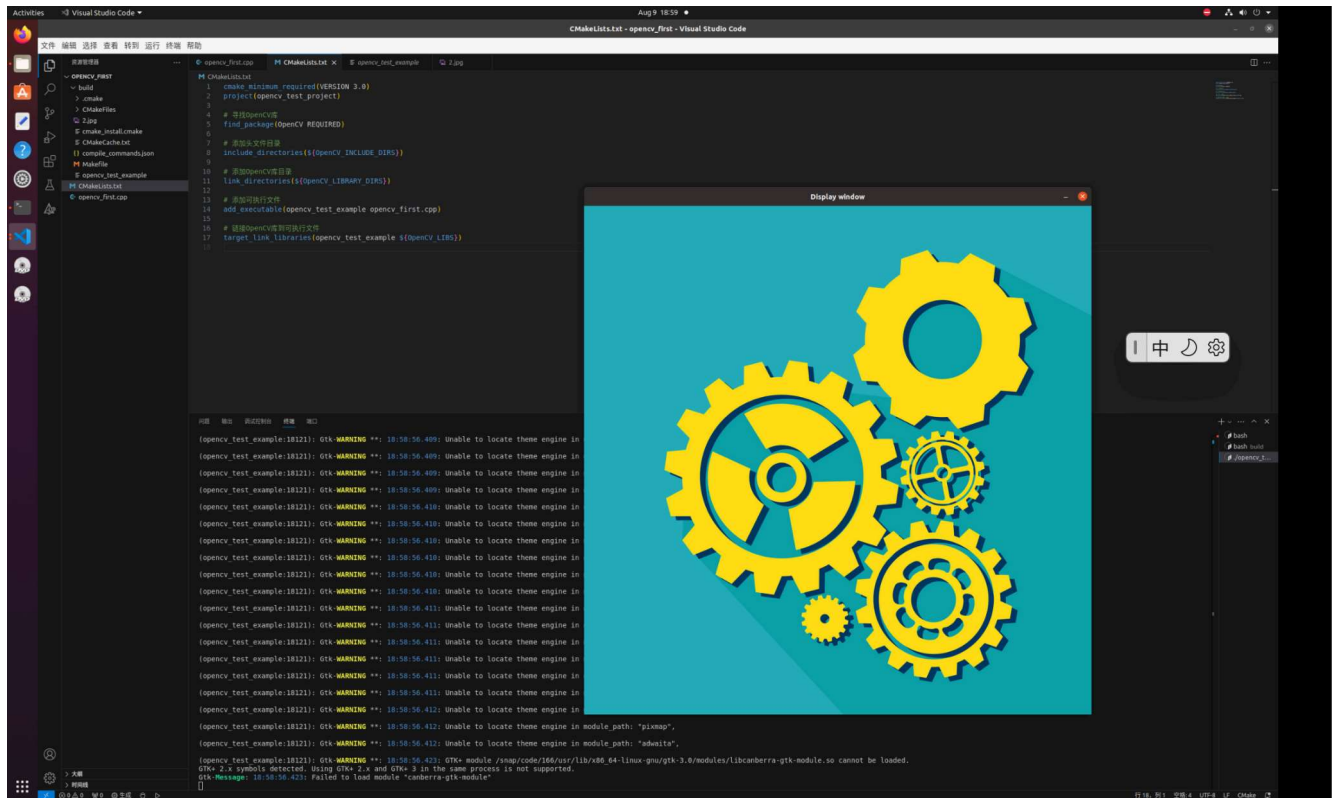


```
1 #include <iostream>
2
3 int main(){
4     std::cout<<"hello"<<std::endl;
5     return 0;
6 }
```

```
ok001@ubuntu:~/program/CProgram/first$ cmake --build build
Scanning dependencies of target example
[ 56%] Building CXX object CMakeFiles/example.dir/first.cpp.o
[100%] Linking CXX executable example
[100%] Built target example
ok001@ubuntu:~/program/CProgram/first$ ./build/example
bash: ./build/example: No such file or directory
ok001@ubuntu:~/program/CProgram/first$ ./build/example
hello
ok001@ubuntu:~/program/CProgram/first$
```

4. opencv

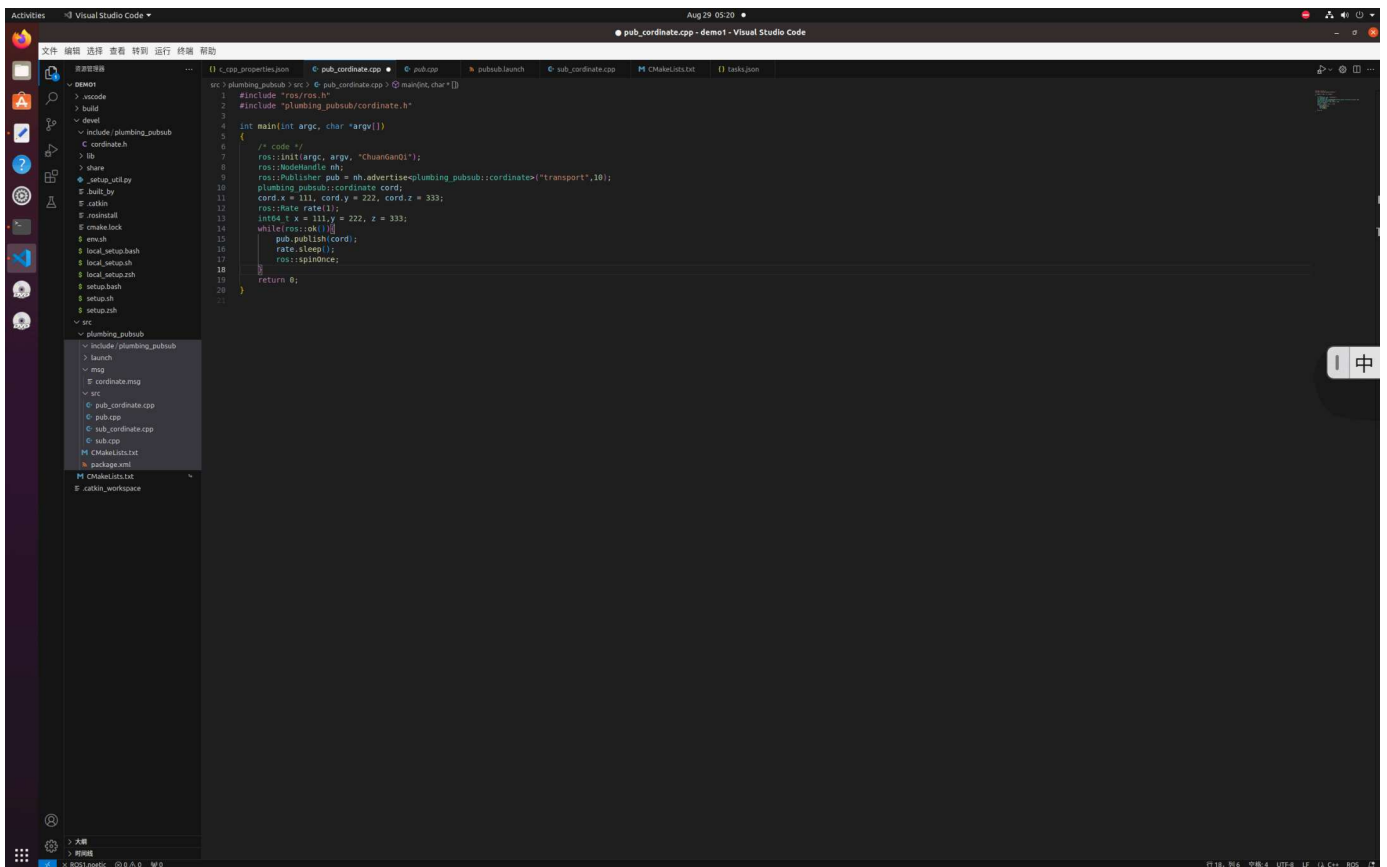
用可执行文件显示图片，源码在另一个文件



P3ROS

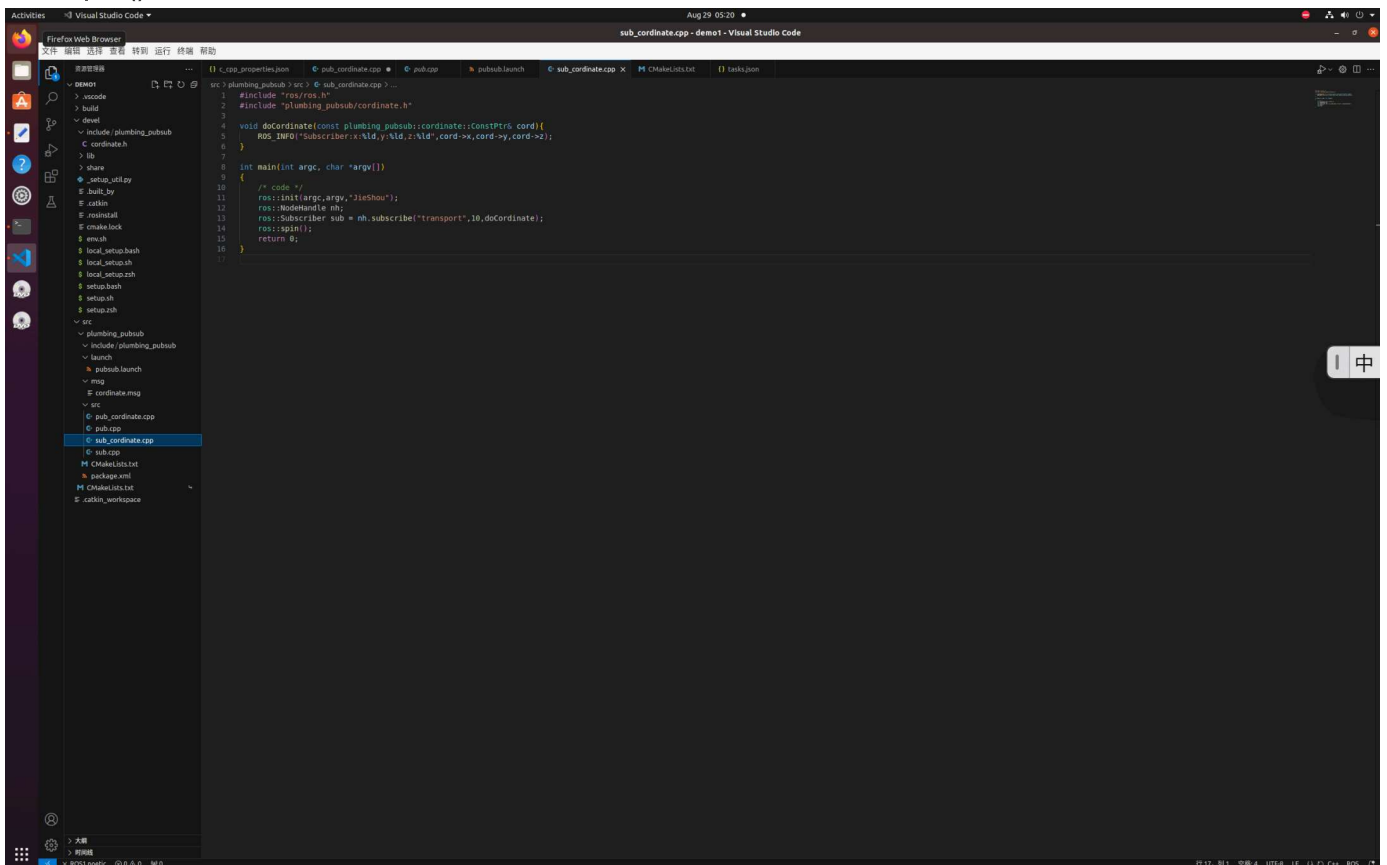
发布者代码：

1. 用msg实现自定义的结构体，并封装在自定义的coordinate.h中。
2. 用NodeHandle变量定义句柄，并写入两变量。其中"transport"为本次通信的名称
3. 循环发布



订阅者代码：

1. 定义接收的回调函数，并在主函数把这个函数传入结构体
2. ros::spin()使不断循环接收，并调用1中定义的回调函数



launch文件及实现接收的效果：

