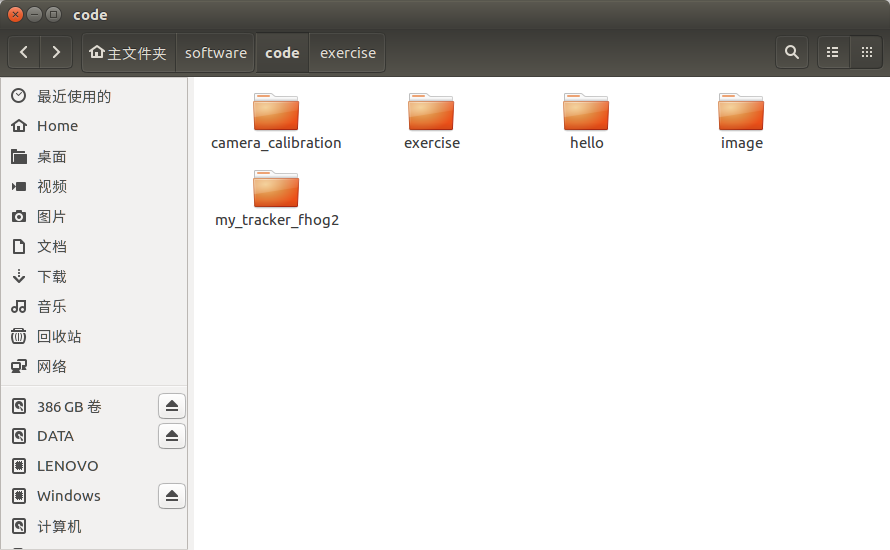
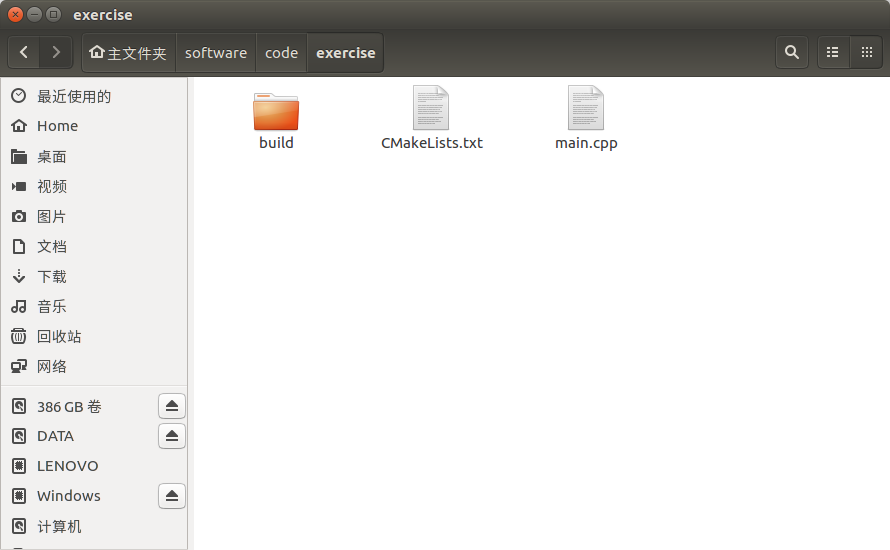
1、编写cmakelists.txt并调试

首先，新建工程文件夹：exercise

2.进入 exercise中

建立main.cpp，CMakeLists.txt和build文件夹



3.编写main.cpp



#include <iostream>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

using namespace std;

using namespace cv;

int main()

{

int a=214324;

a=a+1;

Mat img=imread("/home/xiaweiguang/图片/liushishi3.jpg");

resize(img,img,Size(720,480));

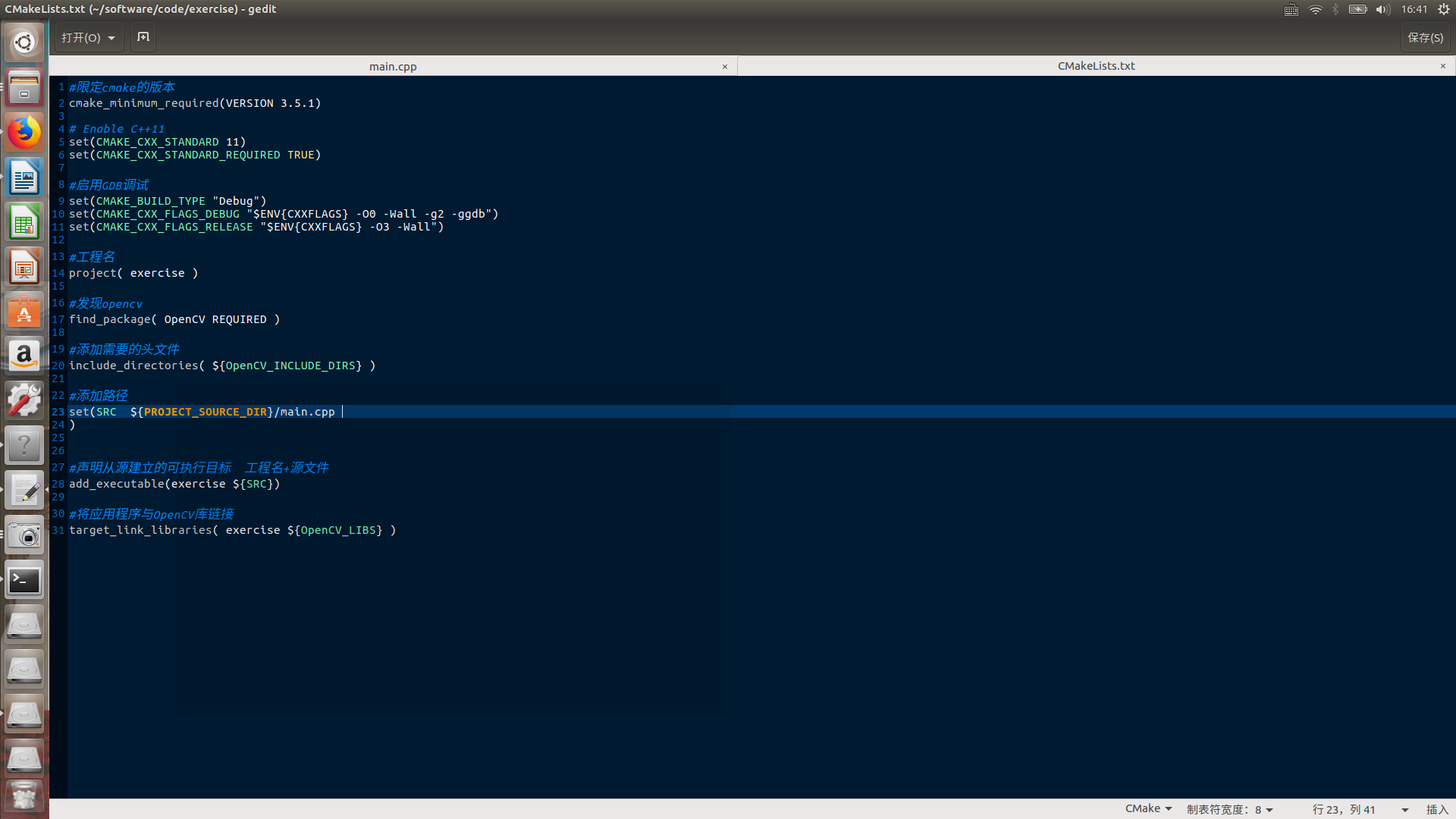
imshow("刘诗诗",img);

waitKey();

return 0;

}

4.编写CMakeLists.txt



第一、限定cmake最低版本 3.5.1

cmake\_minimum\_required(VERSION 3.5.1)

第二、使能C++ 11库

C++11包括大量的新特性：主要特征像[lambda](http://en.cppreference.com/w/cpp/language/lambda)表达式和移动语义，实用的类型推导关键字auto，更简单的容器遍历方法，和大量使模板更容易使用的改进。

set(CMAKE\_CXX\_STANDARD 11)

set(CMAKE\_CXX\_STANDARD\_REQUIRED TRUE)

第三、启用GDB调试

如果想用GDB 进行代码调试需要以下三句话

set(CMAKE\_BUILD\_TYPE "Debug")

set(CMAKE\_CXX\_FLAGS\_DEBUG "$ENV{CXXFLAGS} -O0 -Wall -g2 -ggdb")

set(CMAKE\_CXX\_FLAGS\_RELEASE "$ENV{CXXFLAGS} -O3 -Wall")

第四、工程名

project( exercise )

第五、发现opencv 的包

find\_package( OpenCV REQUIRED )

第六、添加需要的头文件

include\_directories( ${OpenCV\_INCLUDE\_DIRS} )

第七、添加路径

set(SRC ${PROJECT\_SOURCE\_DIR}/main.cpp

//${PROJECT\_SOURCE\_DIR}/src/getAllFiles.hpp)

添加main.cpp的路径，如果还有其他的文件（.h, .hpp, .cpp）可以在exercise下新建src文件夹来存放。

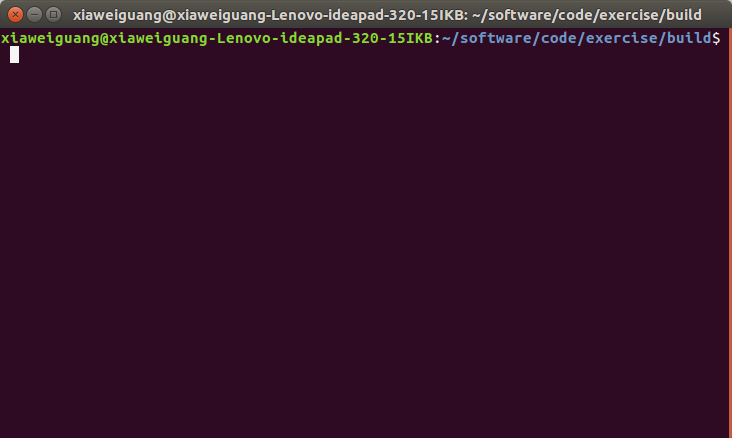
第八、声明从源建立的可执行目标 工程名+源文件

add\_executable(exercise ${SRC})

第九、将应用程序与OpenCV库链接

target\_link\_libraries( exercise ${OpenCV\_LIBS} )

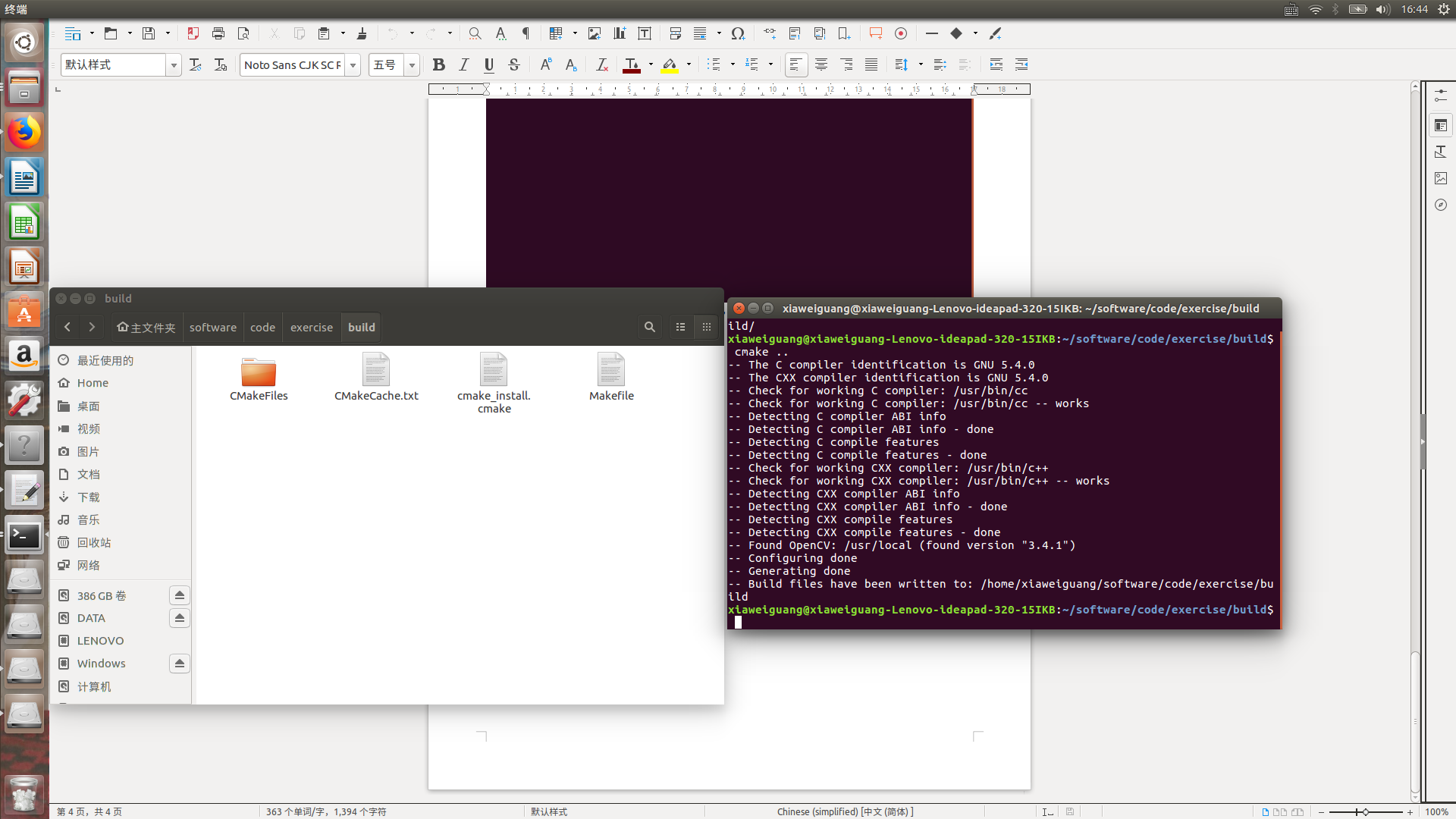
5、开始编译工程

首先，进入exercise/build中(用cd命令进入)

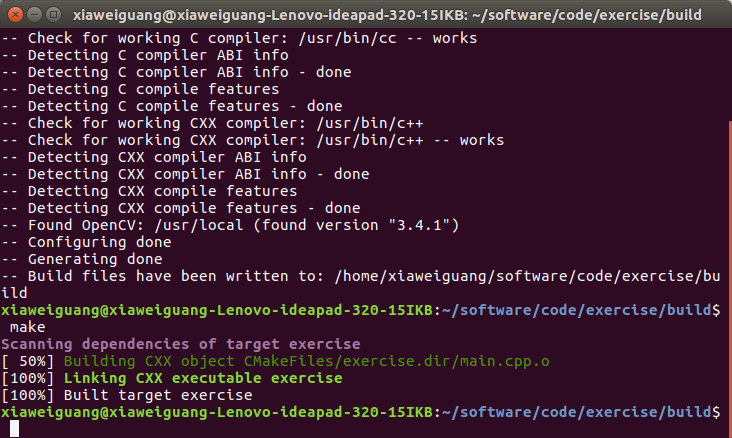
然后，运行

cmake ..

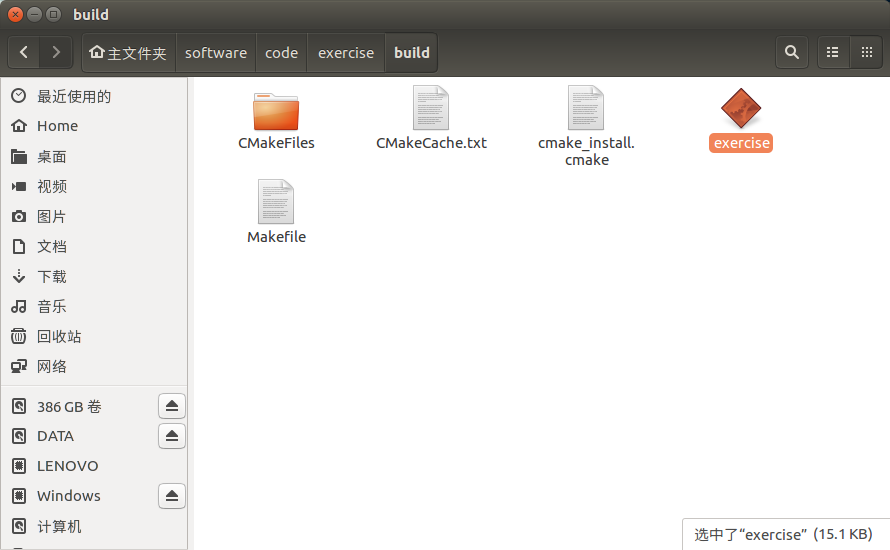
此时build文件夹出现如下文件：

再进行编译

make

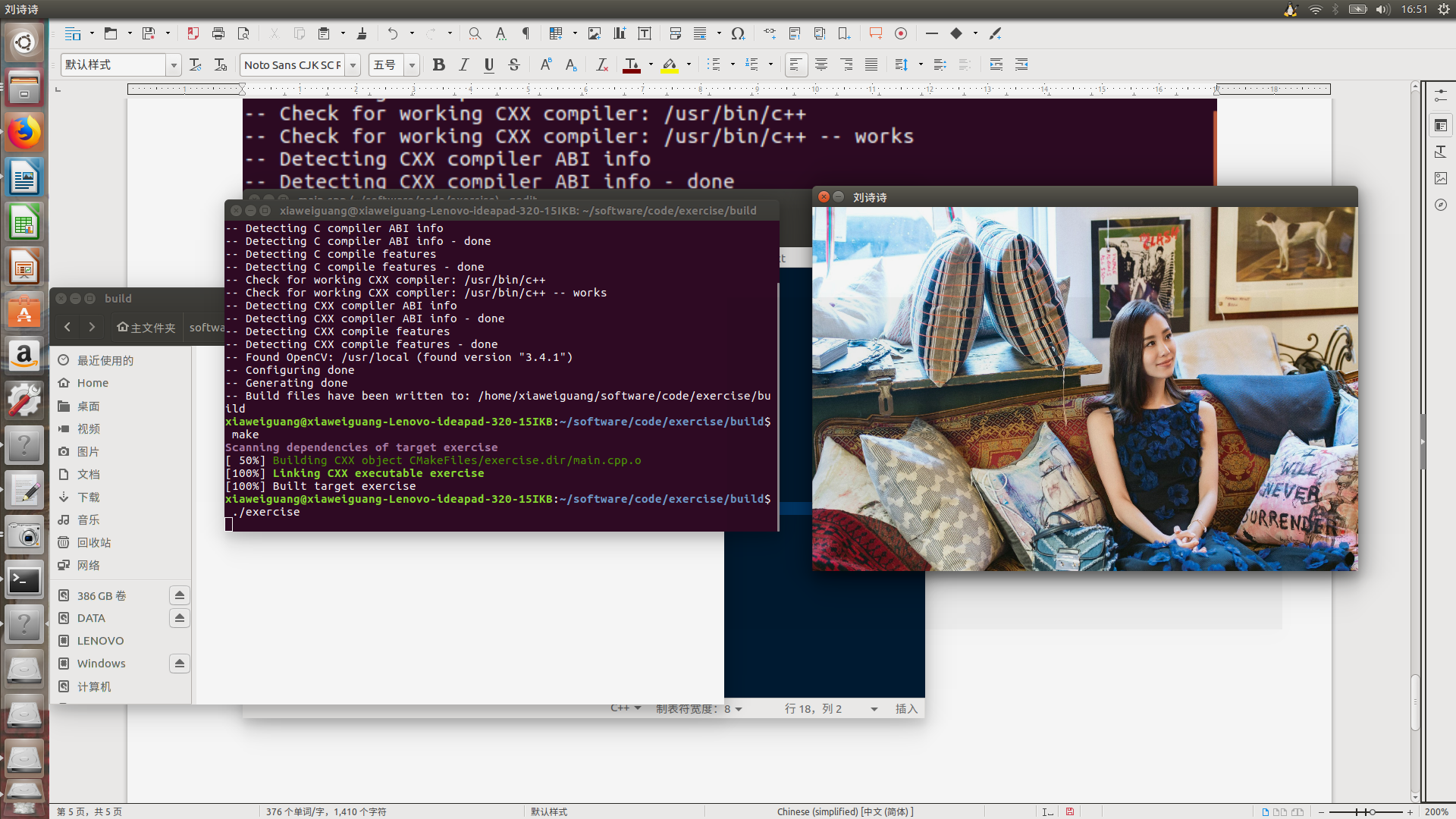
生成可执行文件

exercise，可以双击运行



最后，运行程序

./exercise



6.调试程序(调试可执行程序)

启用GDB 一定要在CMakeLists中加上三句话，

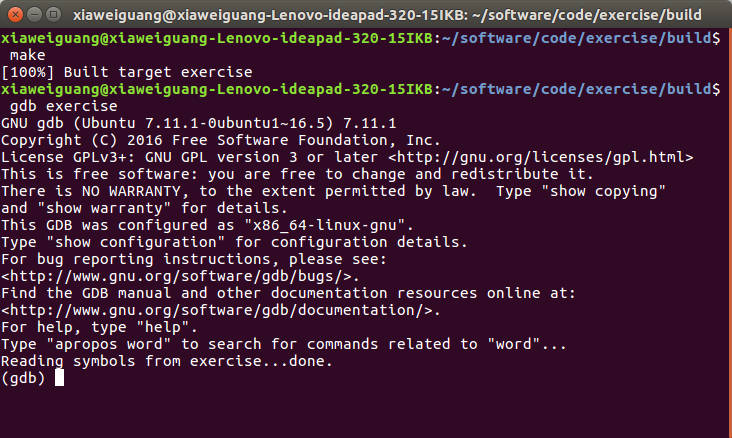
set(CMAKE\_BUILD\_TYPE "Debug")

set(CMAKE\_CXX\_FLAGS\_DEBUG "$ENV{CXXFLAGS} -O0 -Wall -g2 -ggdb")

set(CMAKE\_CXX\_FLAGS\_RELEASE "$ENV{CXXFLAGS} -O3 -Wall")

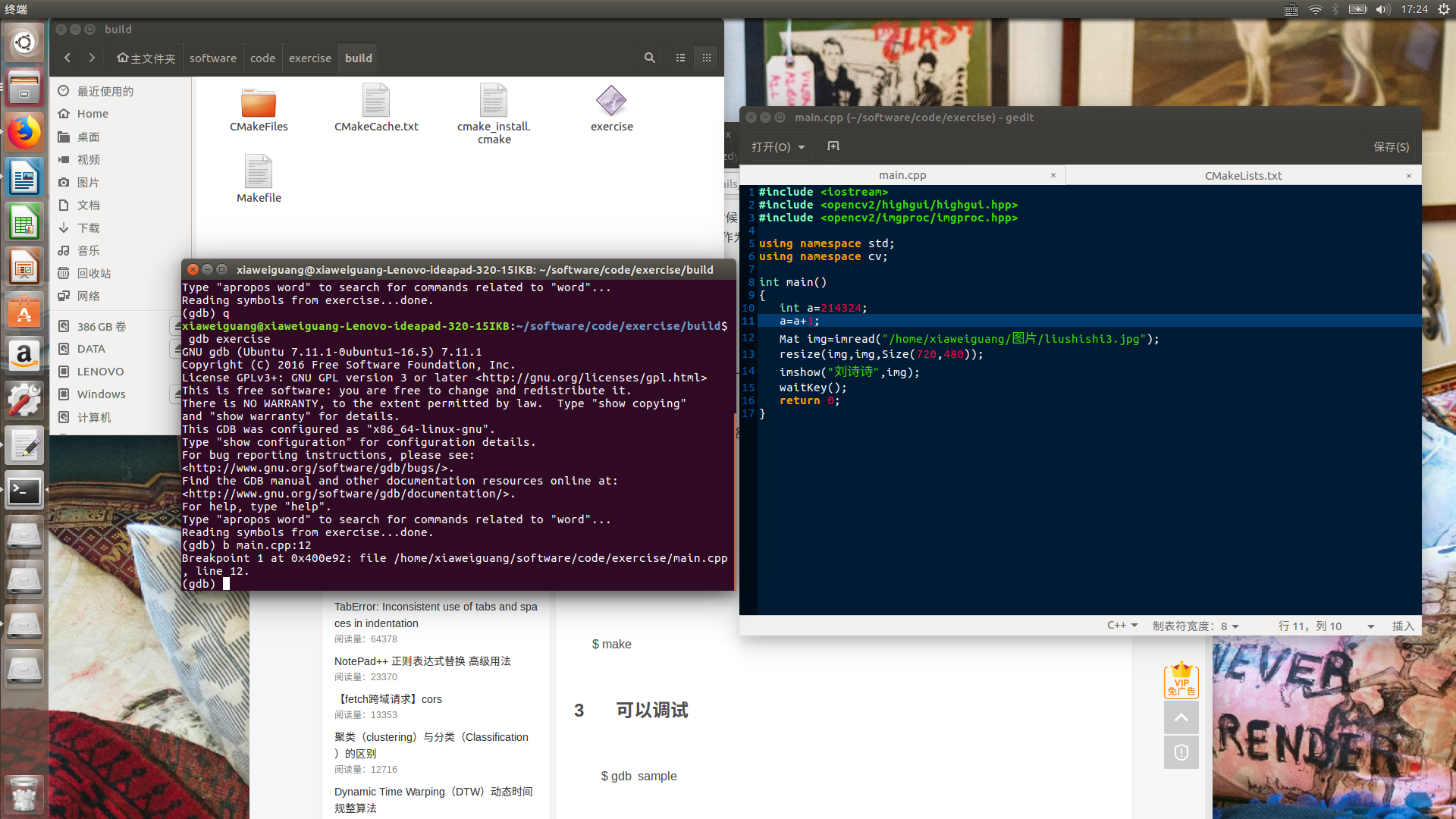
第一、启用GDB

gdb exercise

第二、打断点

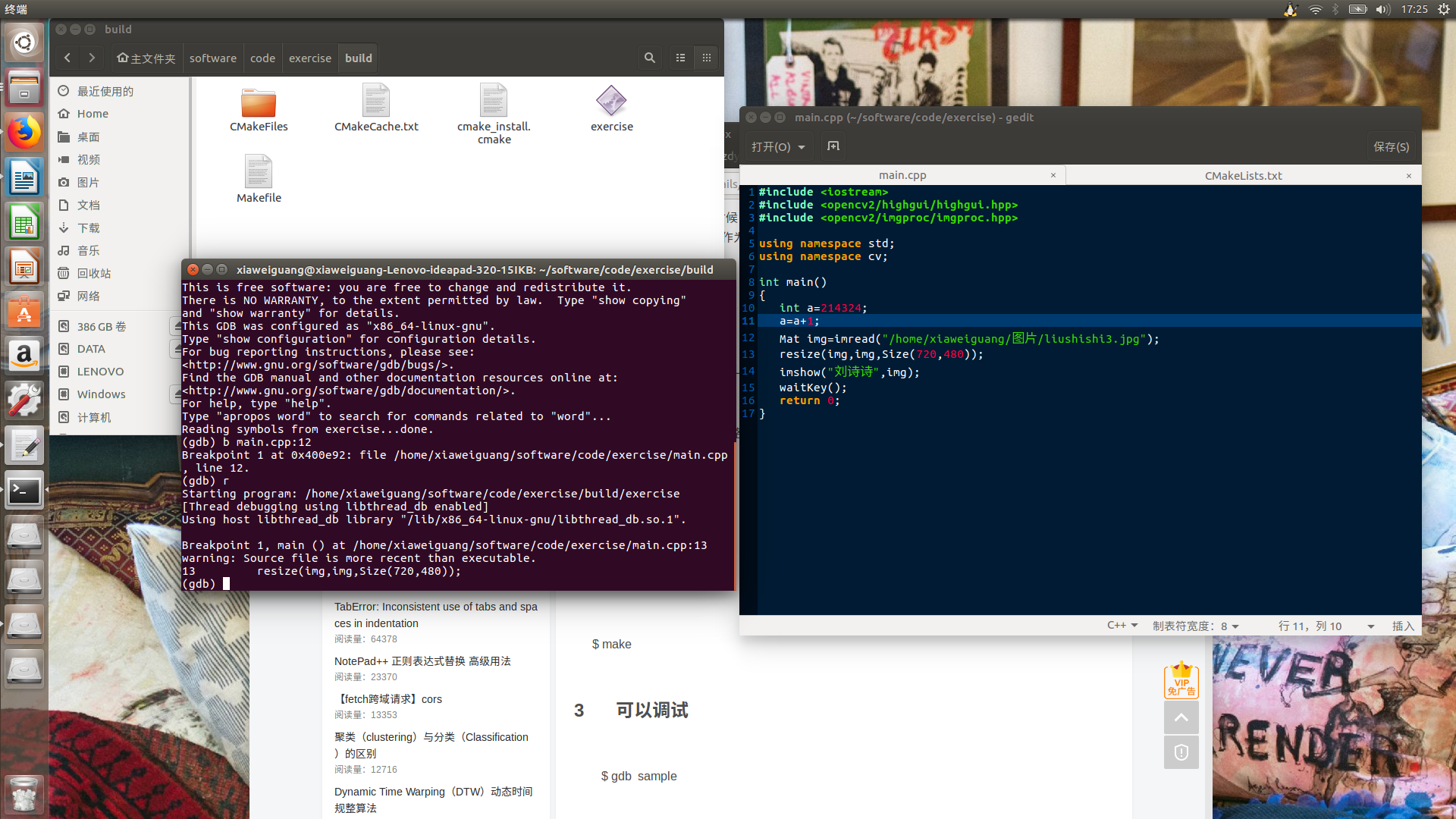
在main.cpp中的第十二行打断点。

b main:10 或者 break main.cpp：12



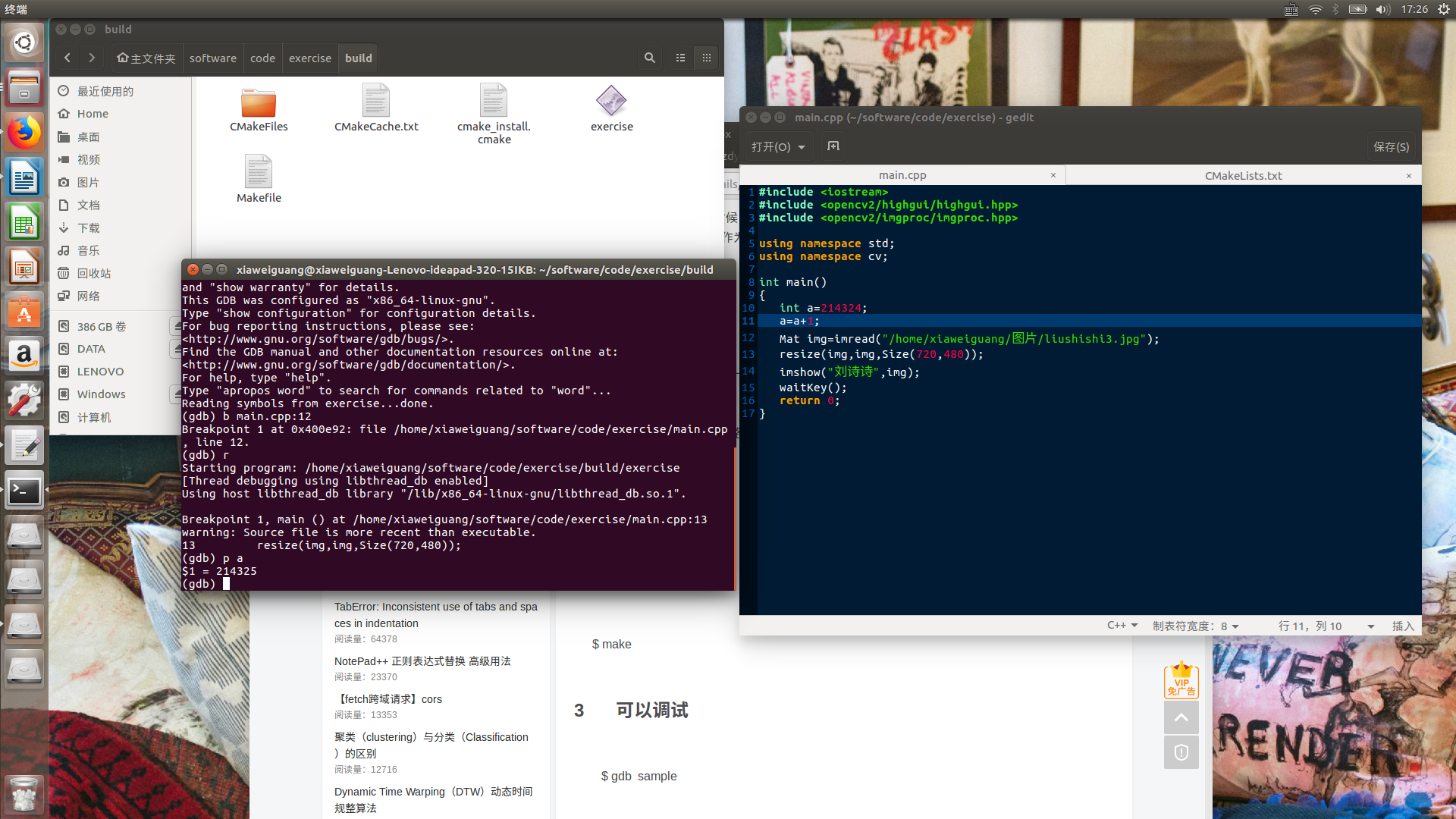
第三、运行

r或者run即可运行程序。



第四、查看变量 a

p a



下面是一些最常用的GDB命令:(可以使用头字母)

       file [filename]  
              装入想要调试的可执行文件

       kill [filename]  
              终止正在调试的程序

       break [file:]function  
              在(file文件的)function函数中设置一个断点

       clear  
              删除一个断点，这个命令需要指定代码行或者函数名作为参数

       run [arglist]  
              运行您的程序 (如果指定了arglist,则将arglist作为参数运行程序)

       bt Backtrace: 显示程序堆栈信息

       print expr  
              打印表达式的值

       continue  
              继续运行您的程序 (在停止之后，比如在一个断点之后)

       list  
              列出产生执行文件的源代码的一部分

       next  
              单步执行 (在停止之后); 跳过函数调用

       nexti  
              执行下一行的源代码中的一条汇编指令

       set  
              设置变量的值。例如：set nval=54 将把54保存到nval变量中

       step  
              单步执行 (在停止之后); 进入函数调用

       stepi  
              继续执行程序下一行源代码中的汇编指令。如果是函数调用，这个命令将进入函数的内部，单步执行函数中的汇编代码

       watch  
              使你能监视一个变量的值而不管它何时被改变

       rwatch  
              指定一个变量，如果这个变量被读，则暂停程序运行，在调试器中显示信息，并等待下一个调试命令。参考rwatch和watch命令

       awatch  
              指定一个变量，如果这个变量被读或者被写，则暂停程序运行，在调试器中显示信息，并等待下一个调试命令。参考rwatch和watch命令

       Ctrl-C  
              在当前位置停止执行正在执行的程序，断点在当前行

       disable  
              禁止断点功能，这个命令需要禁止的断点在断点列表索引值作为参数

       display  
              在断点的停止的地方，显示指定的表达式的值。(显示变量)

       undisplay  
              删除一个display设置的变量显示。这个命令需要将display list中的索引做参数

       enable  
              允许断点功能，这个命令需要允许的断点在断点列表索引值作为参数

       finish  
              继续执行，直到当前函数返回

       ignore  
              忽略某个断点制定的次数。例：ignore 4 23 忽略断点4的23次运行，在第24次的时候中断

       info [name]  
              查看name信息

       load  
              动态载入一个可执行文件到调试器

       xbreak  
              在当前函数的退出的点上设置一个断点

       whatis  
              显示变量的值和类型

       ptype  
              显示变量的类型

       return  
              强制从当前函数返回

       txbreak  
              在当前函数的退出的点上设置一个临时的断点(只可使用一次)

       make  
              使你能不退出 gdb 就可以重新产生可执行文件

       shell  
              使你能不离开 gdb 就执行 UNIX shell 命令

       help [name]  
              显示GDB命令的信息，或者显示如何使用GDB的总体信息

       quit  
              退出gdb.

--------------------- 本文来自 ghostyu 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/ghostyu/article/details/8083228?utm\_source=copy