**代码规范检查作业报告**

**杜旭蕾 2022K8009929003**

**1作业任务概述**

1．生成project文件夹下的三个项目的compile\_commands.json文件。

2．针对其中一个项目使用所给工具对其进行检测，并将结果输出到report目录下，再生成index.html，对每一个检查项生成的Bugs随机确认一个即可，核查其是否为误报。

3．书写测试用例并使用crulet工具单独检测检查项4，看能否发现漏报（不能发现也没有问题）。

**2 bftpd代码规范检查**

**2.1 compile\_commands.json的生成**

生成compile\_commands.json文件的代码如下：

cd ./project

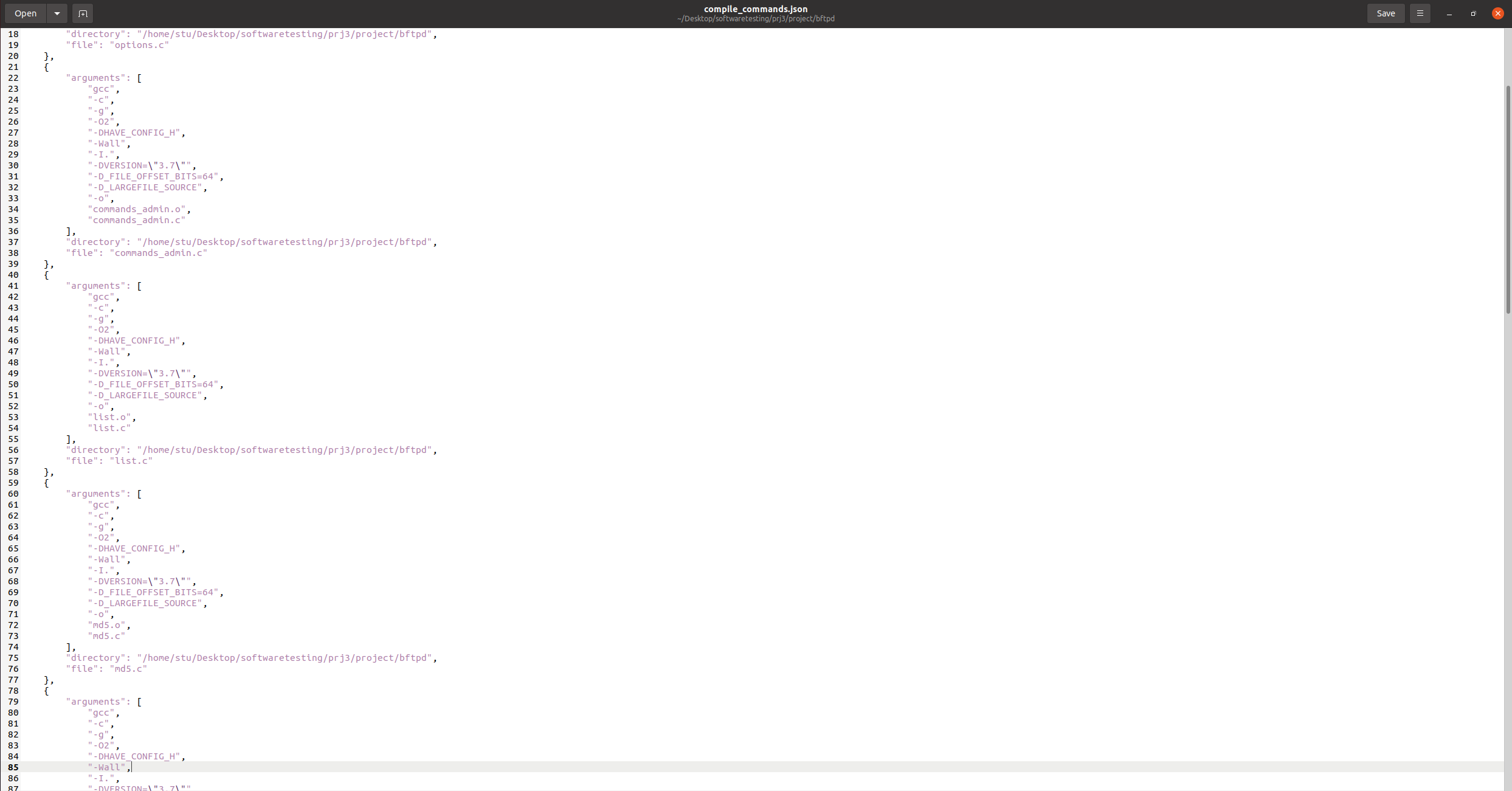
tar -zxvf ./bftpd.tar.gz

chmod -R +111 ./bftpd

cd ./bftpd

bear make

compile\_commands.json的内容如下：



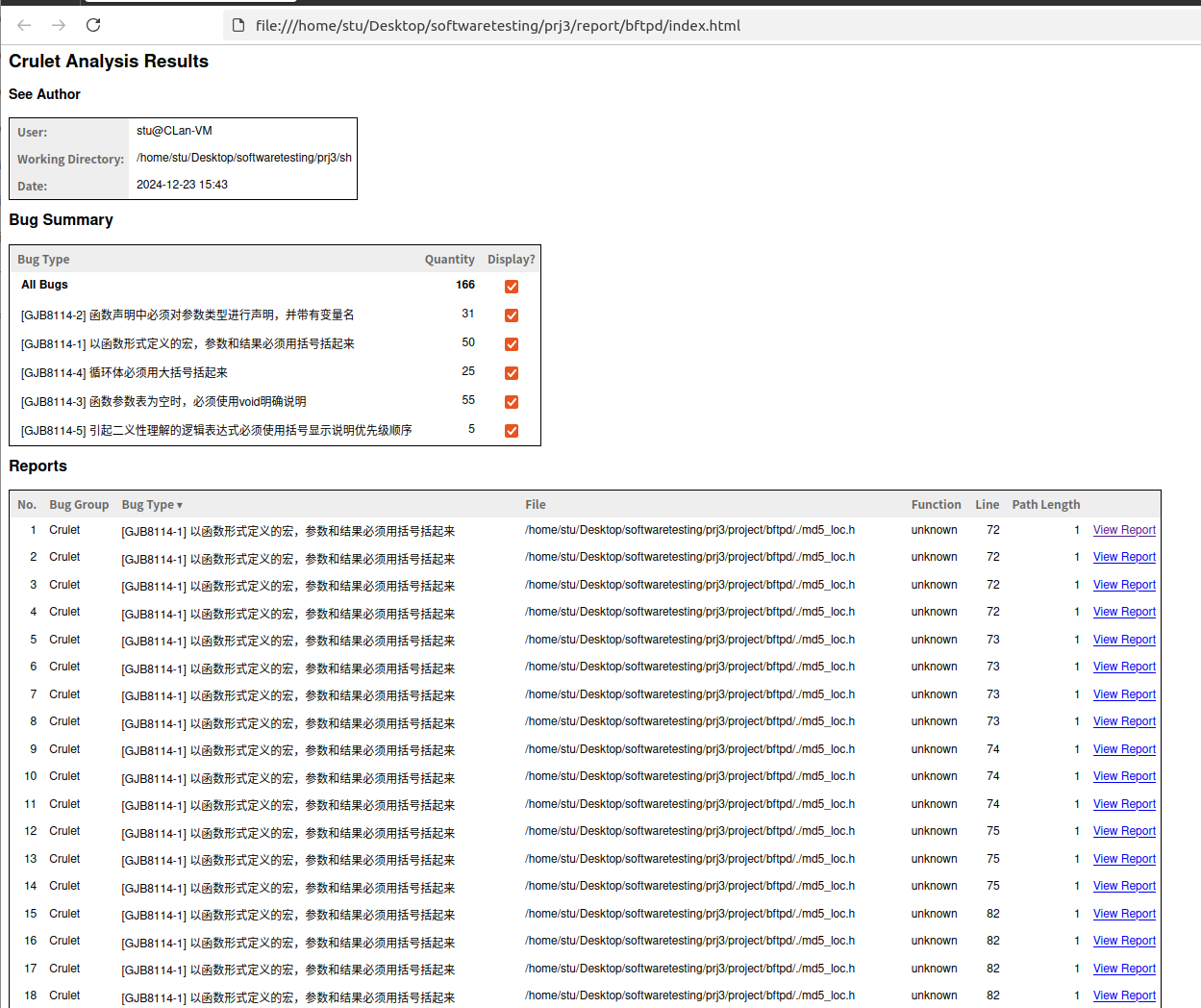
**2.2 index.html文件内容**

生成index.html文件的命令如下：

cd ./sh

bftpd.sh

index.html文件查看如下：

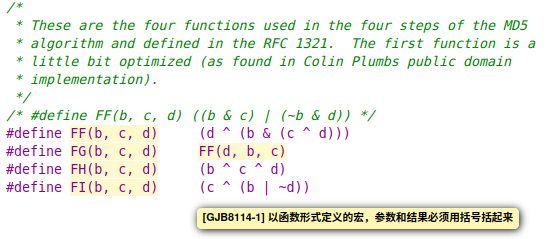


由上图可知，1、2、3、4、5这五类的bug均有共166处，其中3类bug的数量最多为55处，5类bug的数量最少为5处。

**2.3 bug误报检查**

① [GJB8114-1] 以函数形式定义的宏，参数和结果必须用括号括起来

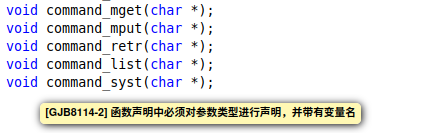
对于1类bug，抽取如下bug进行检查：



发现以上bug中c、b、d参数均未加括号，因此此处不是误报。

② [GJB8114-2] 函数声明中必须对参数类型进行声明，并带有变量名

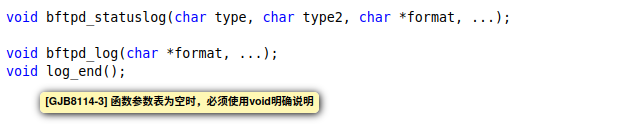
对于2类bug，抽取如下bug进行检查：



以上bug中char\*之后没有变量名，因此此处不是误报。

③ [GJB8114-3] 函数参数表为空时，必须使用void明确说明

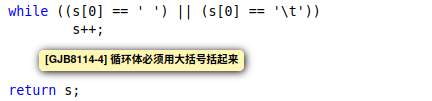
对于3类bug，抽取如下bug进行检查：



以上bug中log\_end函数后应为(void)而不是( )，因此不是误报。

④ [GJB8114-4] 循环体必须用大括号括起来

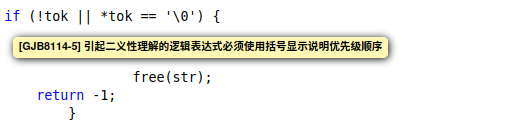
对于4类bug，抽取如下bug进行检查：



在以上bug中由于前面的while语句，s++;行应当被大括号括起来，因此不是误报。

⑤ [GJB8114-5] 引起二义性理解的逻辑表达式必须使用括号显示说明优先级顺序

对于5类bug，抽取如下bug进行检查：



以上bug中，可以将表达式理解为(!tok) || (\*tok == '\0')或者!(tok || \*tok == '\0')，具有二义性，因此不是误报。

**3 celero代码规范检查**

**3.1 compile\_commands.json的生成**

生成compile\_commands.json文件的代码如下：

cd ./project

tar -zxvf ./celero.tar.gz

cd celero

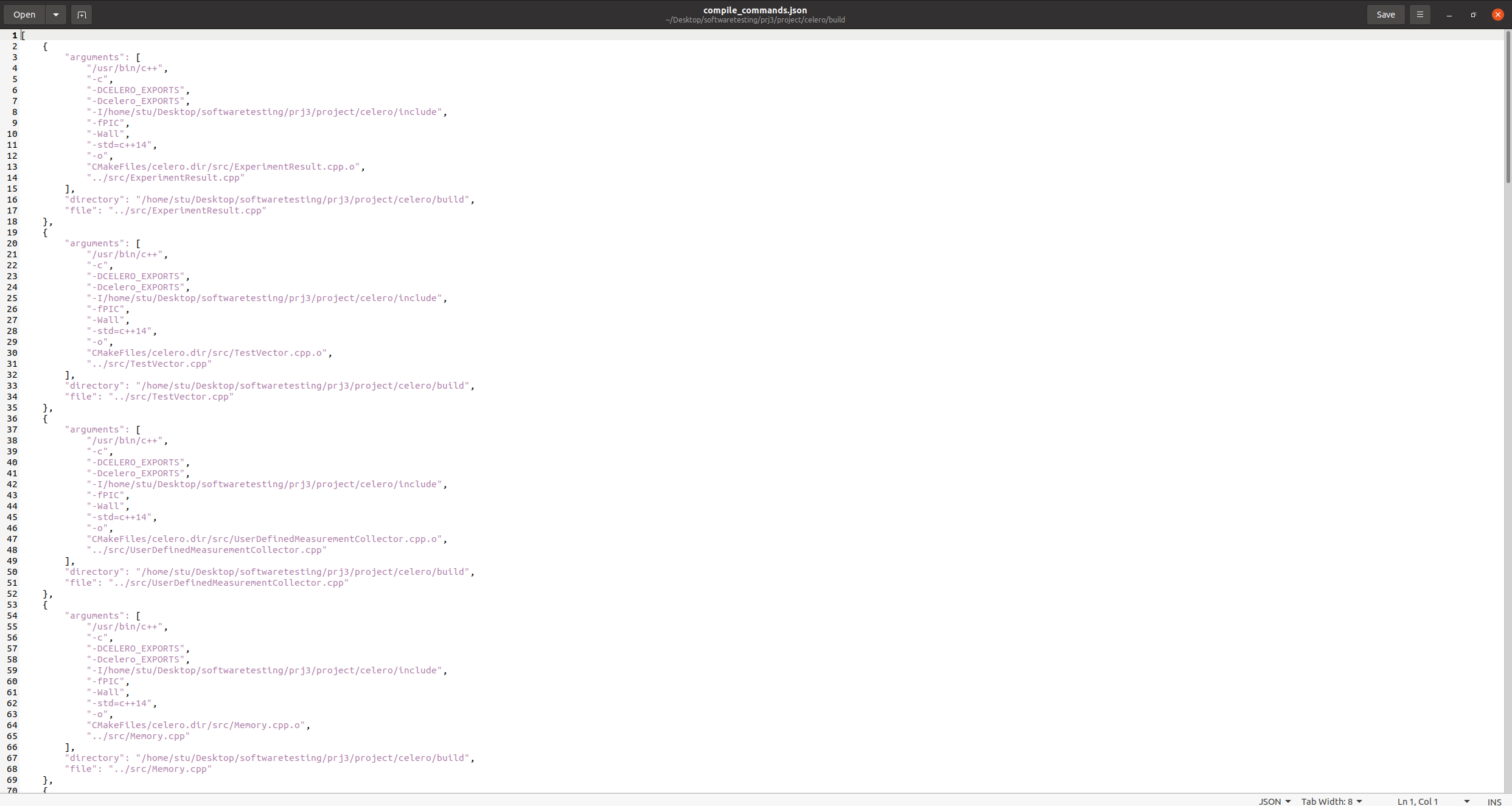
mkdir build

cd ./build

cmake ..

bear make

compile\_commands.json的内容如下：



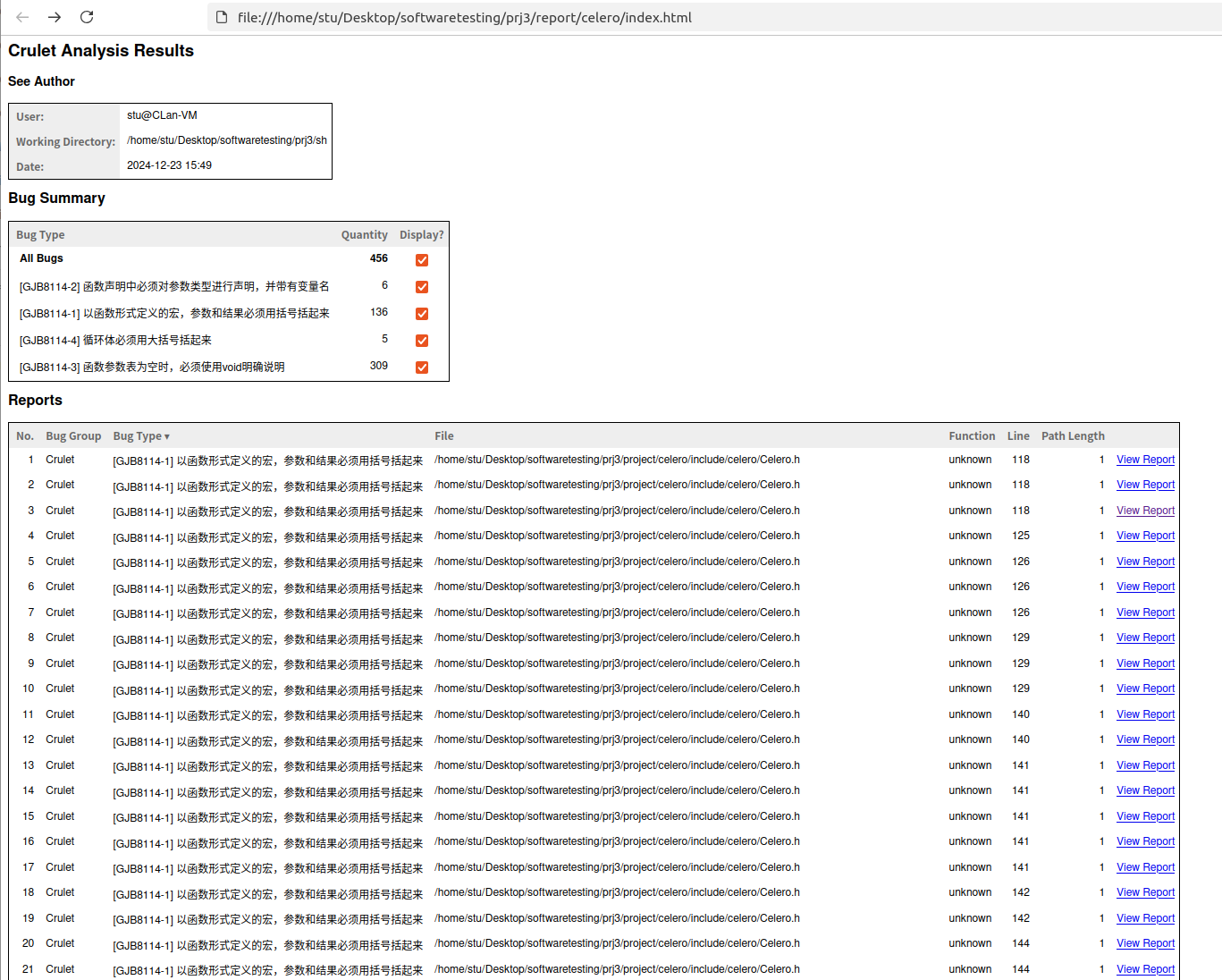
**3.2 index.html文件内容**

生成index.html文件的命令如下：

cd ./sh

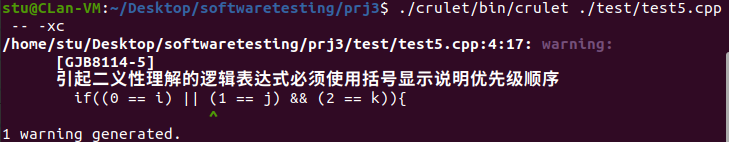
celero.sh

index.html文件查看如下：



由以上结果可以看到本代码只有1、2、3、4类型的bug，一共456条，其中3类型最多，占309条，其次是1类型占136条。没有5类型的bug，因此进行如下检查：

查看test5.cpp：

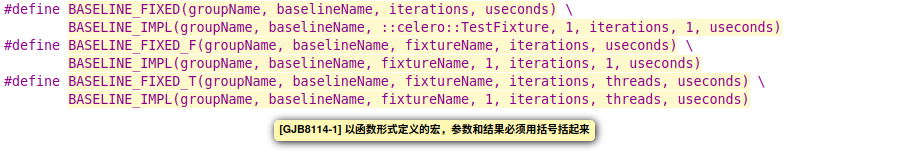


发现报错语句会引起二义性因此没有误报，猜测本次实验代码没有这样的问题。

**3.3 bug误报检查**

① [GJB8114-1] 以函数形式定义的宏，参数和结果必须用括号括起来

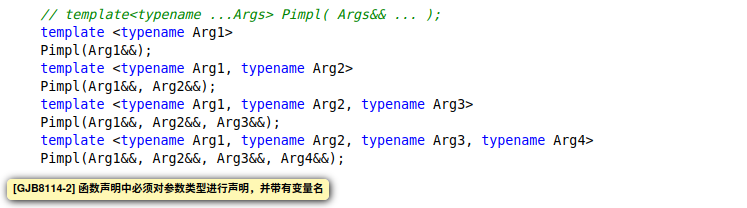
对于1类bug，抽取如下bug进行检查：



由以上可知，在BASELINE\_FIXED\_T这一宏定义的时候没有将使用的变量用括号括起来，没有误报。

② [GJB8114-2] 函数声明中必须对参数类型进行声明，并带有变量名

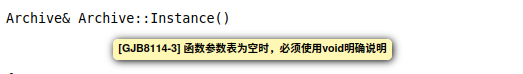
对于2类bug，抽取如下bug进行检查：



以上函数声明中没有参数类型，因此没有误报。

③ [GJB8114-3] 函数参数表为空时，必须使用void明确说明

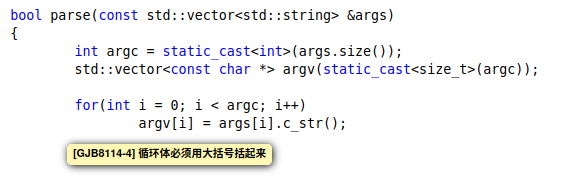
对于3类bug，抽取如下bug进行检查：



由以上内容，可知应当是（void）才对，没有误报。

④[GJB8114-4] 循环体必须用大括号括起来

对于4类bug，抽取如下bug进行检查：



以上函数for后一行应当在{ }里，但是没有，因此没有误报。

**4 cJSON代码规范检查**

**4.1 compile\_commands.json的生成**

生成compile\_commands.json文件的代码如下：

cd ./project

tar -zxvf ./cJSON.tar.gz

cd cJSON

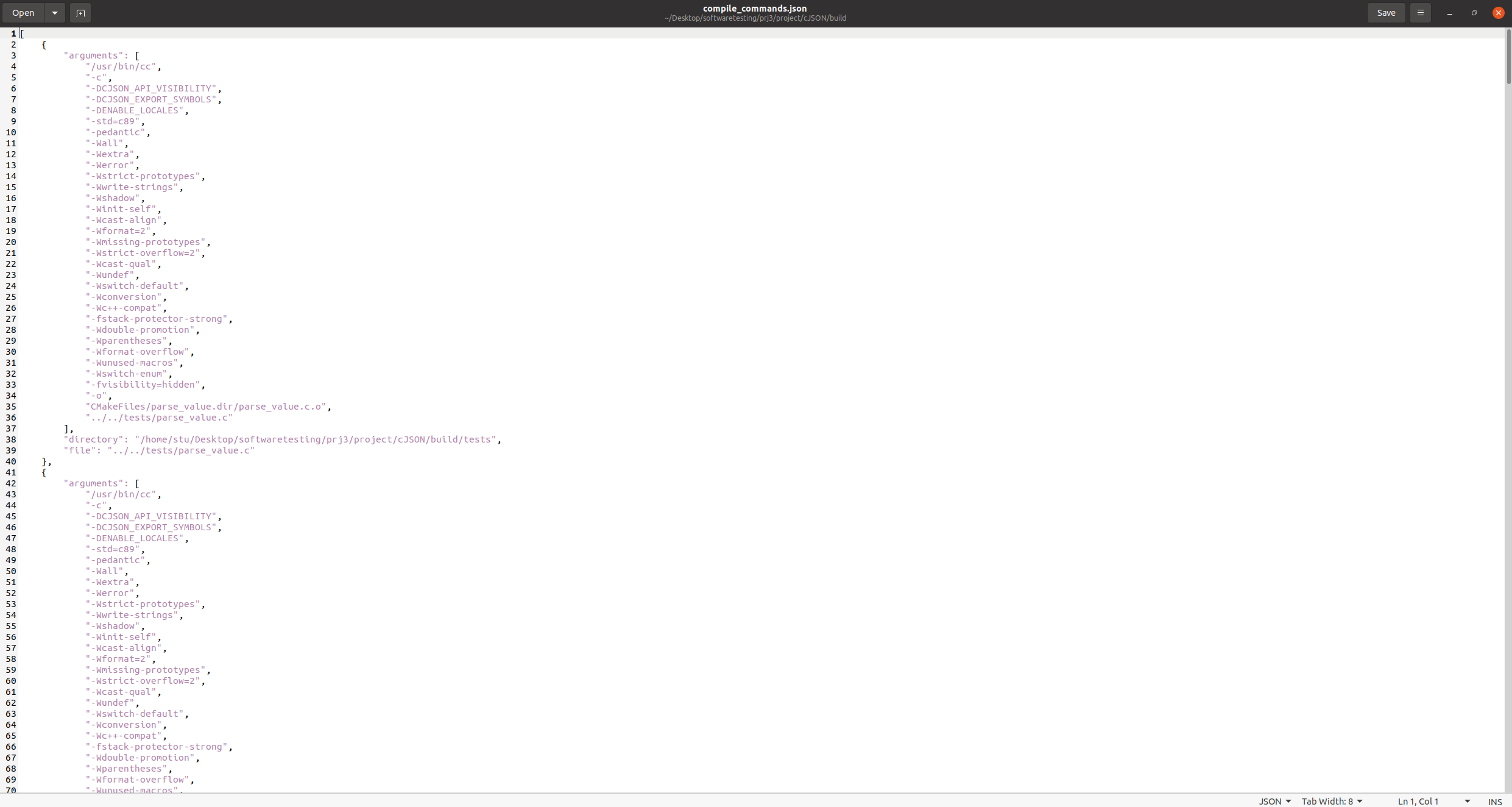
mkdir build

cd ./build

cmake ..

bear make

compile\_commands.json的内容如下：



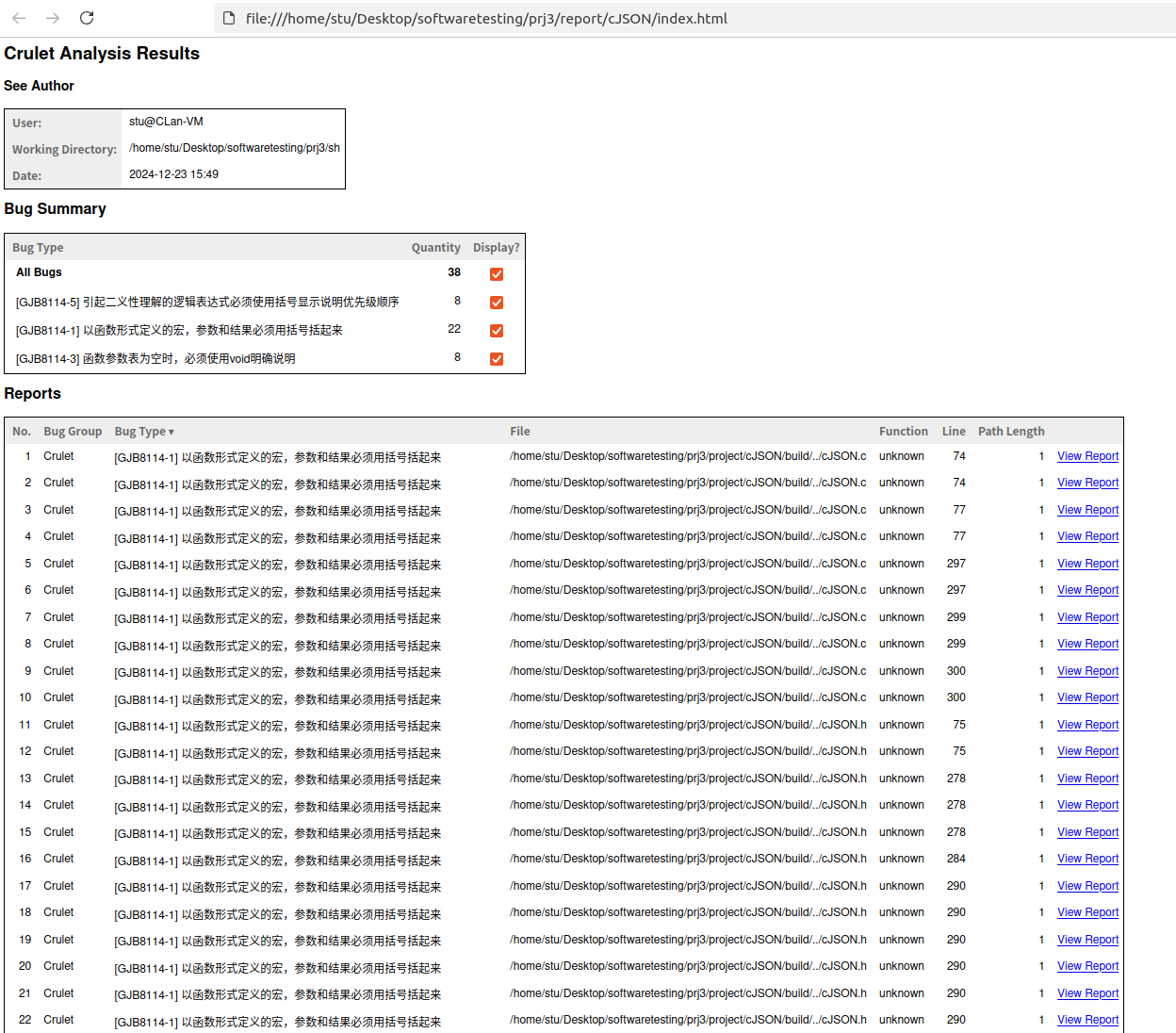
**4.2 index.html文件内容**

生成index.html文件的命令如下：

cd ./sh

cJSON.sh

index.html文件查看如下：

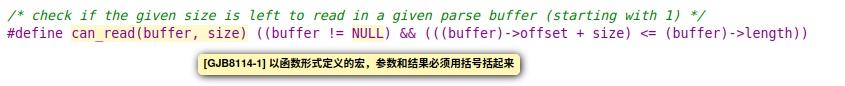


以上bugs中没有2、4类报错，只有1、3、5类bug，总数为38天，其中1类bug最多，占22条。

**4.3 bug误报检查**

① [GJB8114-1] 以函数形式定义的宏，参数和结果必须用括号括起来

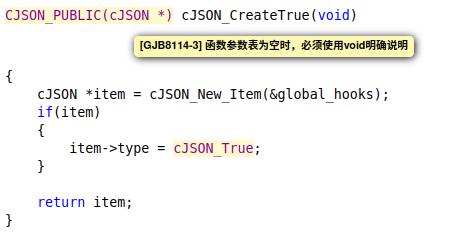
对于1类bug，抽取如下bug进行检查：



以上报错中，定义的语句buffer和size都应当加括号，因此不是误报。

② [GJB8114-3] 函数参数表为空时，必须使用void明确说明

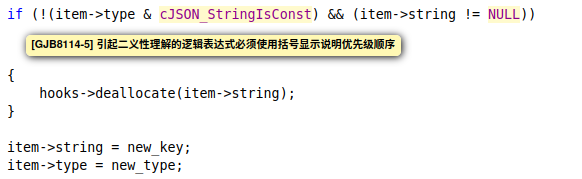
对于3类bug，抽取如下bug进行检查：



以上代码函数参数表确实是void，产生误报。

③ [GJB8114-5] 引起二义性理解的逻辑表达式必须使用括号显示说明优先级顺序

对于5类bug，抽取如下bug进行检查：



以上报错的！符号和&&符号可能会导致二义性的产生，因此不是误报。

**5 书写测试用例的代码规范检查**

**5.1测试用例内容**

书写以下测试用例进行代码规范检查：

#define WHILE(a, b) while(a & b) b--;

int main(void) {

    int i = 0;

    int a = 0;

    int b = 0;

    int c = 0;

    WHILE(a,b);

    while (i < 10)

        while (a && b || c)

            i++;

    return 0;

}

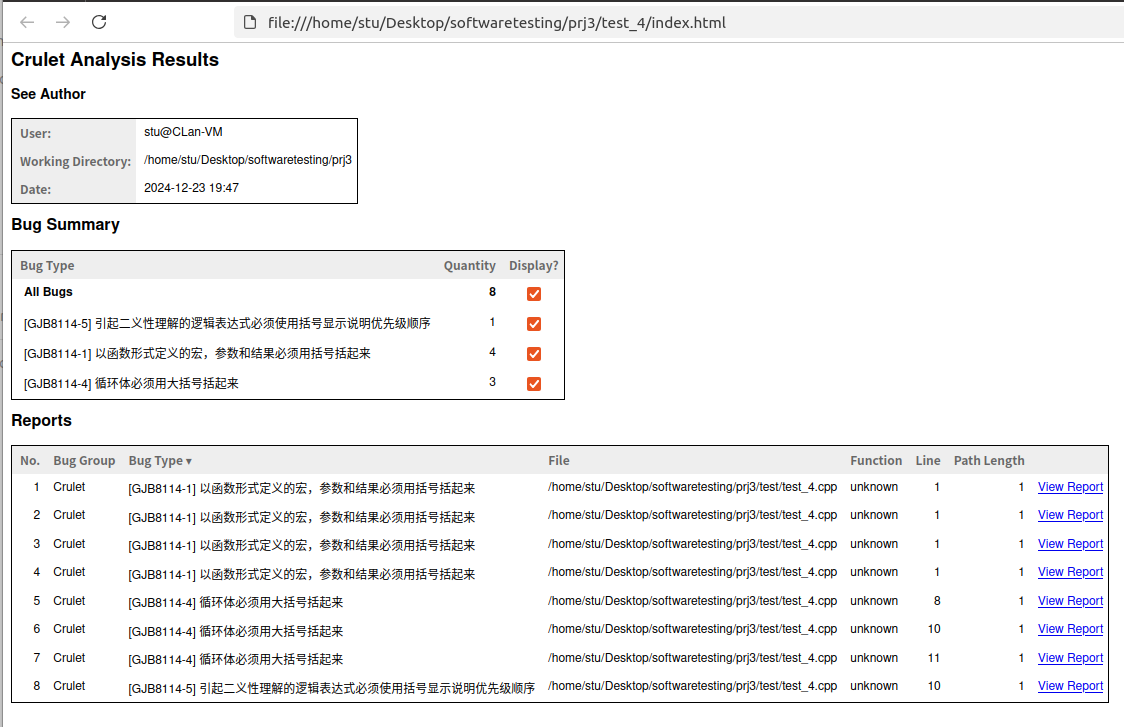
**5.2 index.html文件内容**

生成index.html文件的命令如下：

./crulet/bin/crulet ./test/test\_4.cpp -d ./test\_4 --

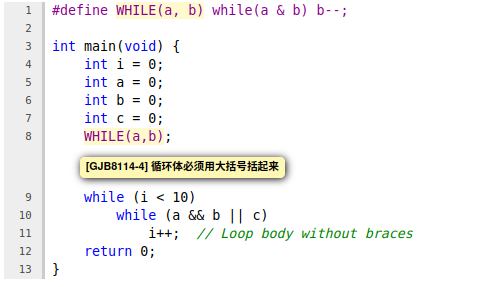
python2 ./genidx/genidx.py ./test\_4

index.html文件查看如下：

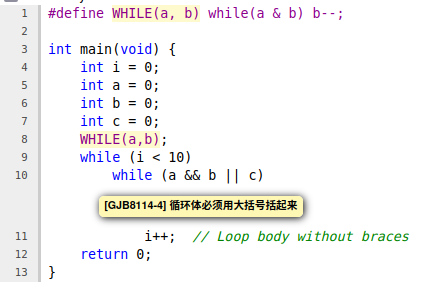


**5.3 bug误报检查**

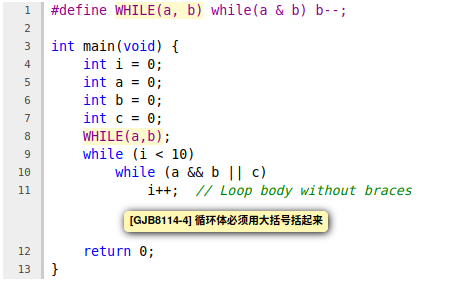
针对类型4的bug（共3个），逐一进行查看如下：



以上是第一处报错，此处调用了WHILE这个宏，宏的内容中没有给while语句后的循环内容加括号，因此不是误报。



以上第二次报错中最外面的循环体没有给内容语句加括号，没有误报。



以上最后一次报错内循环体没有加括号，不是误报。

**6 作业总结**

通过此次代码规范检查作业，我深入了解了如何使用工具对项目代码进行规范性检查，并且熟练掌握了生成 compile\_commands.json 文件以及 index.html 报告的操作。针对不同类型的代码规范问题，对多个项目进行了详细分析，并通过实际例子对误报情况进行了核查，加深了对代码规范性检查这一软件测试的理解。