Student Name: 王研博
Student ID: 2022141460122

高级语言程序设计
Assignment 1

Section 1：Overview of the assignment content

因 docker 故障采用远程 ssh 连接 Ubuntu 虚拟机的方案代替

Section 2：Configure the appropriate docker environment

因 docker 故障采用远程 ssh 连接 Ubuntu 虚拟机的方案代替

Section 3：Compiling single files with g++

3.1: 单步与分步编译 test.cpp 文件，并利用 ls 指令打印出相应的过程文件并截图，执行单步编译得到的 a.out 可执行文件，与分步编译得. 到的 test 可执行文件，给出相应的运行结果

answer:

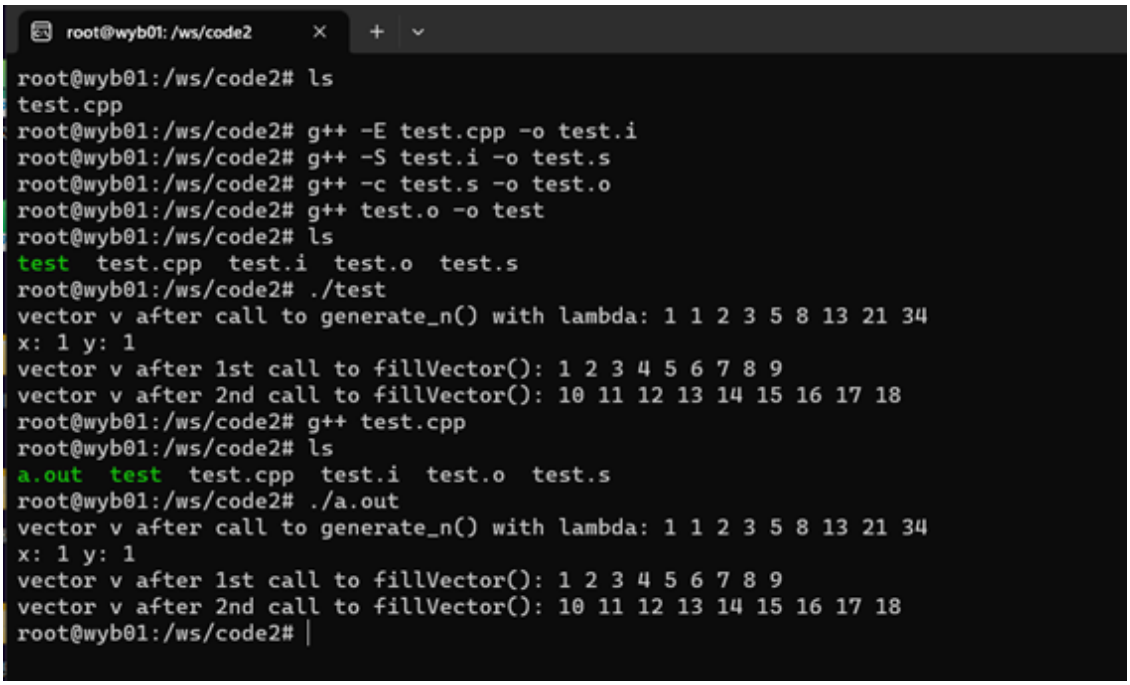

图 1: Single-step and step-by-step compilation

3.2: 给出利用 time 命令打印 inefficency.cpp 优化编译与非优化编译的执行信息

answer:

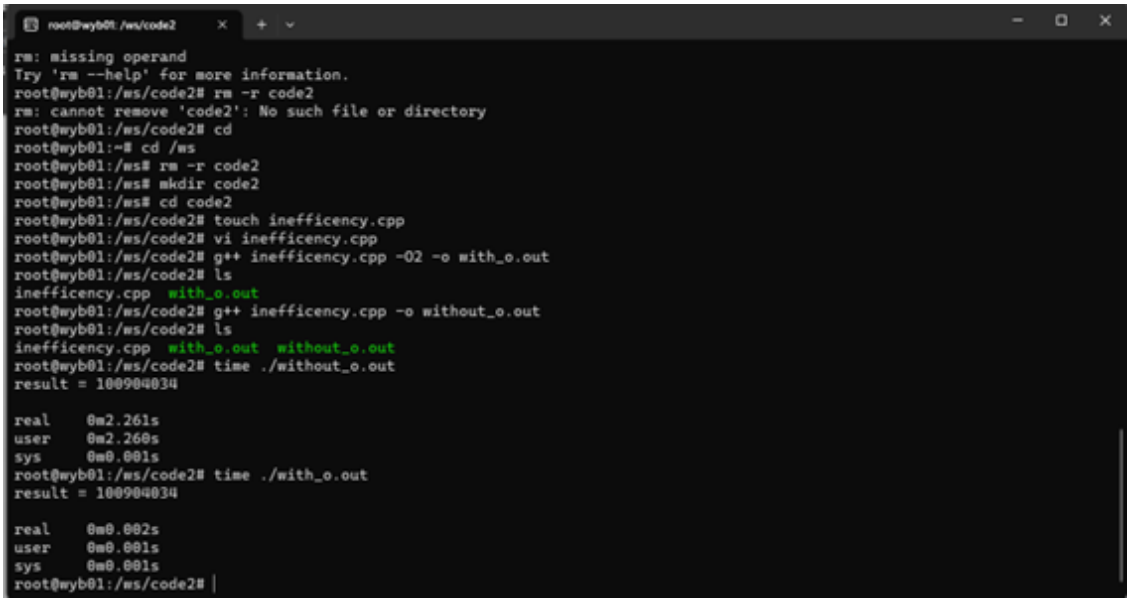

图 2: inefficency.cpp execution information for optimized and non-optimized compilation

3.3: 选择 3 到 4 个其他的 test 文件，尝试利用不同的 g++ 参数进行编译，给出对应生成文件与最终的执行截图

answer:

3.3.1:

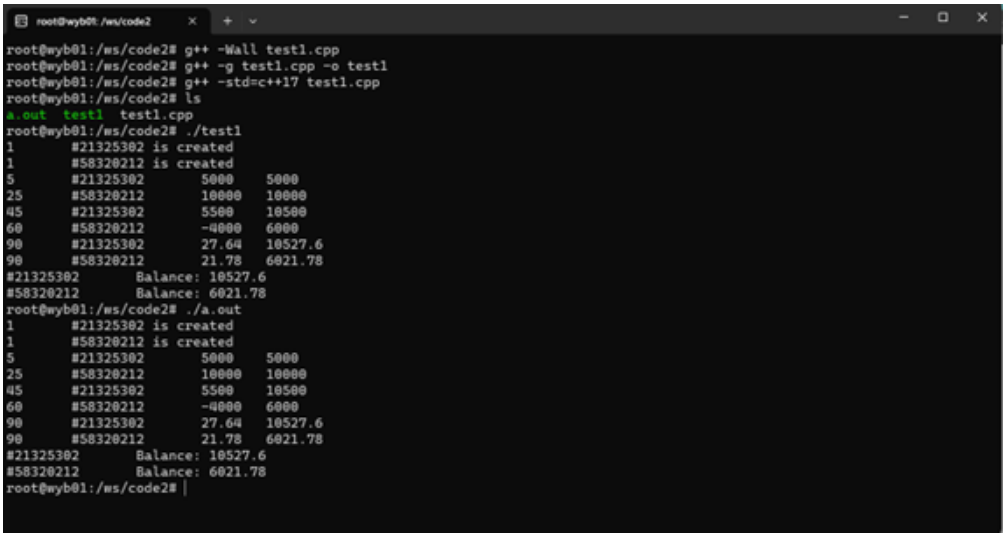

图 3: First:Compile with different g++ parameters and print the final result

3.3.2:

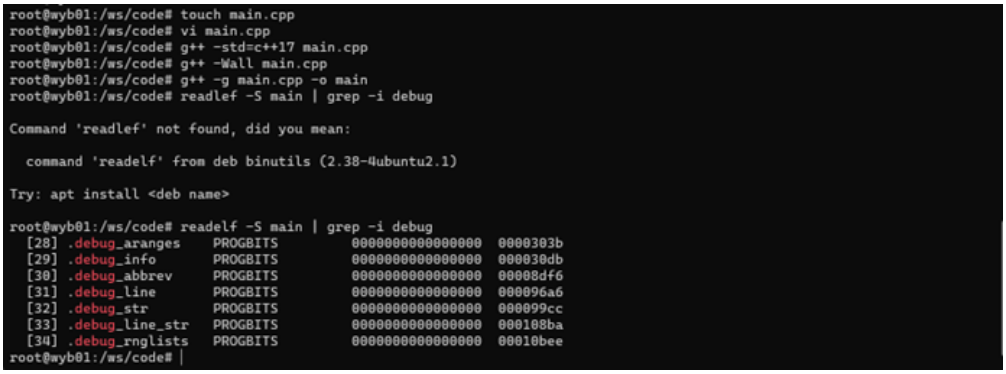

图 4: Second:Compile with different g++ parameters and print the final result

3.3.3:

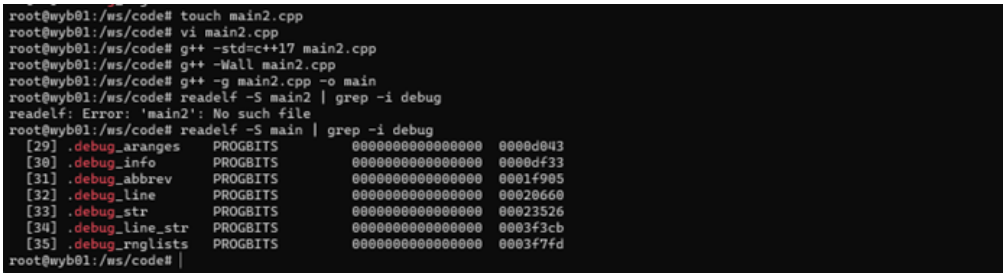

图 5: Third:Compile with different g++ parameters and print the final result

Section 4：Debugging files with GDB

4.1：对于代码片段一，分别追踪前后两次调用函 sumOfSquare() 时函数调用的栈帧与层级关系，并给出过程的相关截图

answer:

4.1.1:

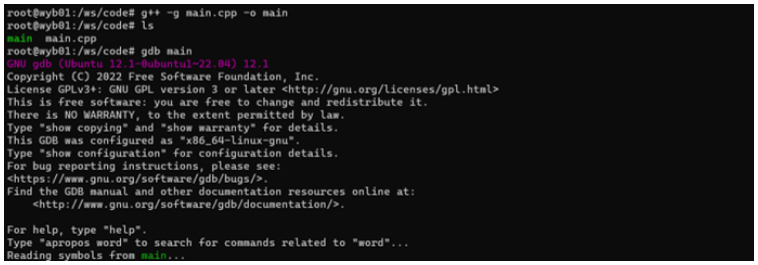

图 6: Compilation and gdb debugging

4.1.2:

```
(gdb) b sumOfSquare
Breakpoint 1 at 0x1257: sumOfSquare. (2 locations)
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   <MULTIPLE>
1.1                         y   0x0000000000001257 in sumOfSquare(int, int) at main.cpp:5
1.2                         y   0x000000000000127b in sumOfSquare(double, double) at main.cpp:9
(gdb) run
```

图 7: Set the breakpoints and run the code

4.1.3:

```
Starting program: /ws/code/main
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter two integer: 2 3

Breakpoint 1, sumOfSquare (a=2, b=3) at main.cpp:5
5               return a * a + b * b;
(gdb) bt
#0  sumOfSquare (a=2, b=3) at main.cpp:5
#1  0x0000555555555329 in main () at main.cpp:16
(gdb) next
6       }
(gdb)
Their sum of square: 13
main () at main.cpp:19
19              cout << "Enter two real number: ";
(gdb)
20              cin >> x >> y;
(gdb)
Enter two real number: 2.2 1.1
21              cout << "Their sum of square: " << sumOfSquare(x, y) << endl;
(gdb) step

Breakpoint 1, sumOfSquare (a=2.2000000000000002, b=1.1000000000000001) at main.cpp:9
9               return a * a + b * b;
(gdb) bt
#0  sumOfSquare (a=2.2000000000000002, b=1.1000000000000001) at main.cpp:9
#1  0x00005555555553b9 in main () at main.cpp:21
```

图 8: View the stack frame and hierarchy relationship on two function calls separately (backtrace command)

4.2：对于代码片段二，我们希望您能够追踪每次循环的变量 y 的具体变量值，并思考最后打印的 j 数值的意义

answer:

4.2.1:

```
(gdb) b 9
Note: breakpoint 1 also set at pc 0x555555555534e.
Breakpoint 2 at 0x555555555534e: file range_based_for.cpp, line 9.
(gdb) run
Starting program: /ws/code2/test
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 2, main () at range_based_for.cpp:9
9               for( int y : x ) { // Access by value using a copy declared as a specific type.
(gdb) display y
1: y = 0
(gdb) next
11                      cout << y << " ";
1: y = 1
(gdb)
9               for( int y : x ) { // Access by value using a copy declared as a specific type.
1: y = 1
(gdb)
11                      cout << y << " ";
1: y = 2
(gdb)
9               for( int y : x ) { // Access by value using a copy declared as a specific type.
1: y = 2
(gdb)
11                      cout << y << " ";
1: y = 3
(gdb)
```

图 9: Set a breakpoint to monitor the change in y: the first loop

4.2.2:

图 10: Set a breakpoint to monitor the change in y: the second loop

4.2.3:

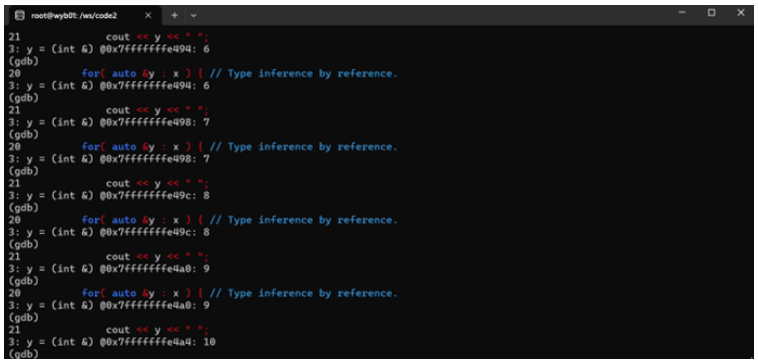

图 11: Set a breakpoint to monitor the change in y: the third loop(Passing by reference)

4.2.4:

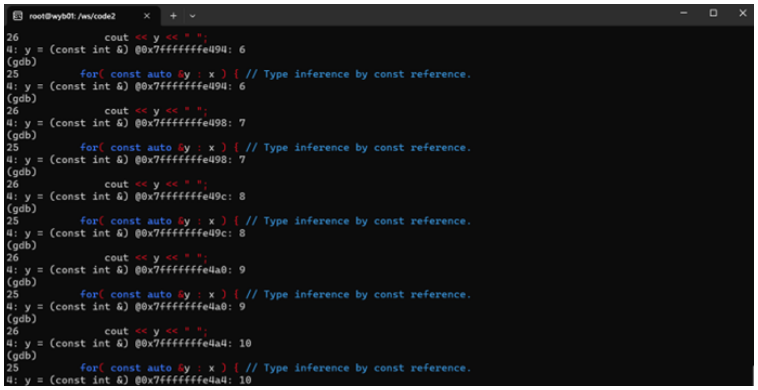

图 12: Set a breakpoint to monitor the change in y: the forth loop(Passing by reference)

4.2.5:

图 13: Print the value of each j

SUM: The meaning of j is to print out the data stored in the vector container

4.3：代码片段三，我们希望您能根据调试与代码中的提示修改代码让其正常运行，并告诉我们 const，enum，define 三者中哪一个有地址，并将其地址打印出来。

answer:

图 14: Print the address of the three

Conclusion:

in this problem enum does not have an address, const and define do

My interpretation：

After setting the breakpoint, you can see that p is replaced with ”hello” in Pre-processing, so the value printed by define p has the address

c.NUM is a static const int type, and you can find the address printed out.

But C::NUM1 is an enum type, it doesn’t print out the address and output: Can’t take address of ”C::NUM1” which isn’t an lvalue.