

Lab Report for Object-oriented Programming
course
Lab 2: Preprocessor

Wang, Chen
16307110064
School of Software
Fudan University

April 17, 2019

Contents

1	Background Knowledge & Concepts Required for This Lab	2
1.1	C/C++ Compiling Process	2
1.1.1	Overall process of compiling	2
1.1.2	Preprocessing	2
1.1.3	Parsing	2
1.1.4	Global Optimization	2
1.1.5	Code Generation	2
1.1.6	Peehole Optimization	2
1.1.7	Linking	2
1.2	C/C++ Preprocessing	2
1.2.1	The need of preprocessing	2
1.2.2	Different preprocessing algorithms	2
1.2.3	Preprocessing algorithm utilized by the current g++	2
1.2.4	Encapsulation	2
2	Specifications of This Lab	3
2.1	Regulations in the preprocessing process	3
2.2	Test cases designed in the lab	3
2.3	Other specifications of the programming	3
3	Structure and the OO Ideas Adopted	4
3.1	Objected-oriented ideas adopted in the implementation	4
3.1.1	Encapsulation	4
4	Running Result of My Implementation	5
4.1	The first sample test	5

Chapter 1

Background Knowledge & Concepts Required for This Lab

1.1 C/C++ Compiling Process

1.1.1 Overall process of compiling

1.1.2 Preprocessing

1.1.3 Parsing

1.1.4 Global Optimization

1.1.5 Code Generation

1.1.6 Peehole Optimization

1.1.7 Linking

1.2 C/C++ Preprocessing

1.2.1 The need of preprocessing

1.2.2 Different preprocessing algorithms

1.2.3 Preprocessing algorithm utilized by the current g++

1.2.4 Encapsulation

Chapter 2

Specifications of This Lab

- 2.1 Regulations in the preprocessing process
- 2.2 Test cases designed in the lab
- 2.3 Other specifications of the programming

Chapter 3

Structure and the OO Ideas Adopted

3.1 Objected-oriented ideas adopted in the im- plementation

3.1.1 Encapsulation

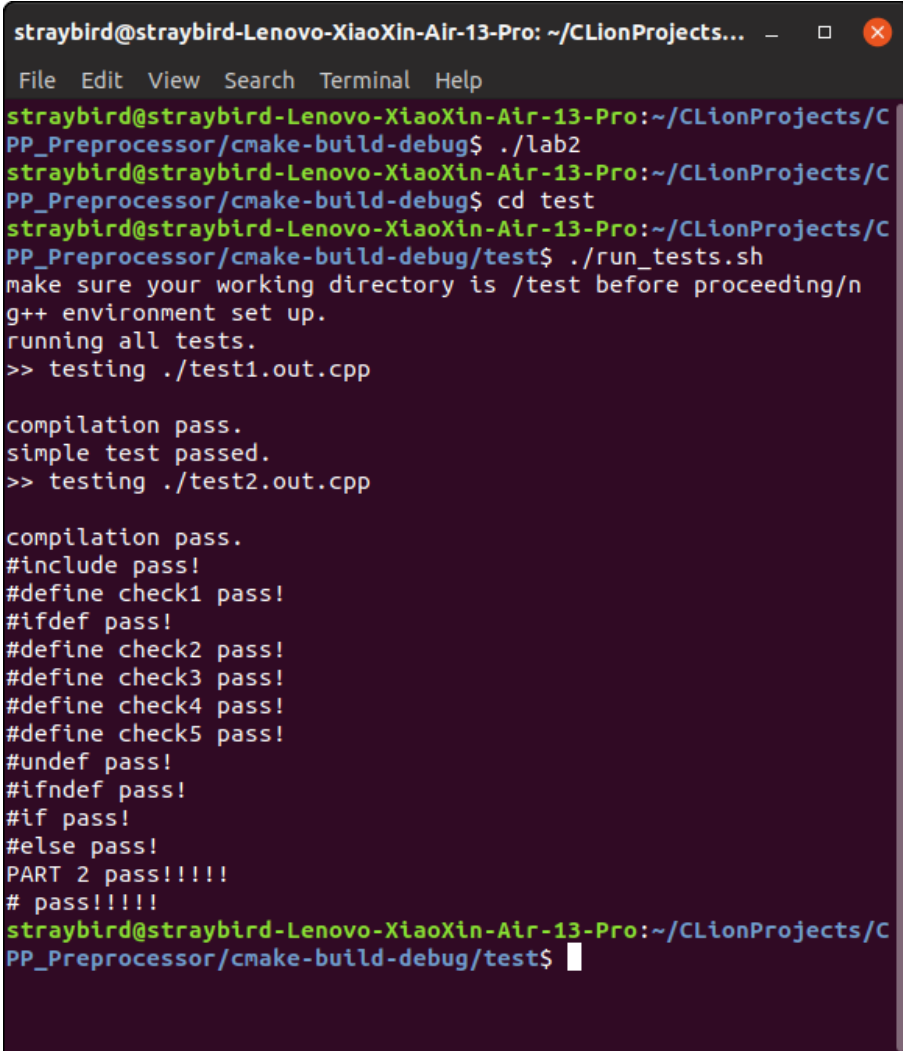
Chapter 4

Running Result of My Implementation

The following screenshots are the tests that are identical to the steps in the requirement documentation and proves that my version of implementation functions identical to the standard version.

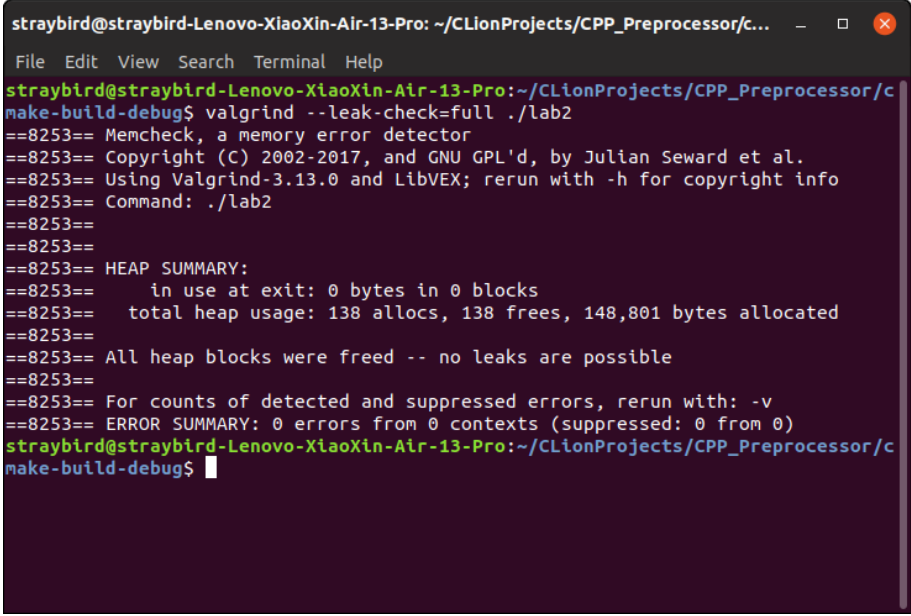
4.1 The first sample test

The results are shown as Figure 4.2.

A terminal window with a dark background and light-colored text. The window title is "straybird@straybird-Lenovo-XiaoXin-Air-13-Pro: ~/CLionProjects...". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows a series of commands and their outputs. The user navigates to a directory and runs a script. The script outputs instructions about the working directory and environment, then runs two tests. The first test, "test1.out.cpp", passes compilation and a simple test. The second test, "test2.out.cpp", passes compilation and a series of preprocessor checks, including "#include pass!", "#define check1 pass!", "#ifdef pass!", "#define check2 pass!", "#define check3 pass!", "#define check4 pass!", "#define check5 pass!", "#undef pass!", "#ifndef pass!", "#if pass!", "#else pass!", "PART 2 pass!!!!", and "# pass!!!!". The terminal ends with the prompt "straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/CPP_Preprocessor/cmake-build-debug/test\$".

```
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro: ~/CLionProjects/...  
File Edit View Search Terminal Help  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/C  
PP_Preprocessor/cmake-build-debug$ ./lab2  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/C  
PP_Preprocessor/cmake-build-debug$ cd test  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/C  
PP_Preprocessor/cmake-build-debug/test$ ./run_tests.sh  
make sure your working directory is /test before proceeding/n  
g++ environment set up.  
running all tests.  
>> testing ./test1.out.cpp  
  
compilation pass.  
simple test passed.  
>> testing ./test2.out.cpp  
  
compilation pass.  
#include pass!  
#define check1 pass!  
#ifdef pass!  
#define check2 pass!  
#define check3 pass!  
#define check4 pass!  
#define check5 pass!  
#undef pass!  
#ifndef pass!  
#if pass!  
#else pass!  
PART 2 pass!!!!  
# pass!!!!  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/C  
PP_Preprocessor/cmake-build-debug/test$
```

Figure 4.1: Testcase Result

A terminal window with a dark background and light-colored text. The window title is "straybird@straybird-Lenovo-XiaoXin-Air-13-Pro: ~/CLionProjects/CPP_Preprocessor/c...". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows a command prompt "straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/CPP_Preprocessor/cmake-build-debug\$" followed by the command "valgrind --leak-check=full ./lab2". The output is a series of lines from Valgrind, including version information, copyright, and a heap summary. The summary states that all heap blocks were freed and no leaks are possible. The terminal ends with the command prompt "straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/CPP_Preprocessor/cmake-build-debug\$".

```
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro: ~/CLionProjects/CPP_Preprocessor/c...  
File Edit View Search Terminal Help  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/CPP_Preprocessor/c  
make-build-debug$ valgrind --leak-check=full ./lab2  
==8253== Memcheck, a memory error detector  
==8253== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.  
==8253== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info  
==8253== Command: ./lab2  
==8253==  
==8253==  
==8253== HEAP SUMMARY:  
==8253==     in use at exit: 0 bytes in 0 blocks  
==8253==   total heap usage: 138 allocs, 138 frees, 148,801 bytes allocated  
==8253==  
==8253== All heap blocks were freed -- no leaks are possible  
==8253== For counts of detected and suppressed errors, rerun with: -v  
==8253== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)  
straybird@straybird-Lenovo-XiaoXin-Air-13-Pro:~/CLionProjects/CPP_Preprocessor/c  
make-build-debug$
```

Figure 4.2: Memory Leak Check

Bibliography

- [1] Wikipedia contributors. (2019, February 3). Encapsulation (computer programming). In *Wikipedia, The Free Encyclopedia*. Retrieved 10:19, March 23, 2019, from [https://en.wikipedia.org/w/index.php?title=Encapsulation_\(computer_programming\)&oldid=881507936](https://en.wikipedia.org/w/index.php?title=Encapsulation_(computer_programming)&oldid=881507936)
- [2] Wikipedia contributors. (2019, March 17). Reversi. In *Wikipedia, The Free Encyclopedia*. Retrieved 10:20, March 23, 2019, from <https://en.wikipedia.org/w/index.php?title=Reversi&oldid=888167585>
- [3] Wikipedia contributors. (2019, March 15). Polymorphism (computer science). In *Wikipedia, The Free Encyclopedia*. Retrieved 10:21, March 23, 2019, from [https://en.wikipedia.org/w/index.php?title=Polymorphism_\(computer_science\)&oldid=887878749](https://en.wikipedia.org/w/index.php?title=Polymorphism_(computer_science)&oldid=887878749)
- [4] Wikipedia contributors. (2019, February 27). Object-oriented programming. In *Wikipedia, The Free Encyclopedia*. Retrieved 10:22, March 23, 2019, from https://en.wikipedia.org/w/index.php?title=Object-oriented_programming&oldid=885274966
- [5] Wikipedia contributors. (2019, February 21). Inheritance (object-oriented programming). In *Wikipedia, The Free Encyclopedia*. Retrieved 10:22, March 23, 2019, from [https://en.wikipedia.org/w/index.php?title=Inheritance_\(object-oriented_programming\)&oldid=884436146](https://en.wikipedia.org/w/index.php?title=Inheritance_(object-oriented_programming)&oldid=884436146)