**Lab 3.1 Using splint for C static analysis**

**Overview**

The learning objective of this lab is for students to gain the first-hand experience on using static code analysis tools to check c program for security vulnerabilities and coding mistakes.

Splint [link](http://www.splint.org/) is a tool for statically checking C programs for security vulnerabilities and programming mistakes. Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information.

11 kinds of problems detected by Splint include:

* Dereferencing a possibly null pointer;
* Using possibly undefined storage or returning storage that is not properly defined;
* Type mismatches, with greater precision and flexibility than provided by C compilers;
* Violations of information hiding;
* Memory management errors including uses of dangling references and memory leaks;
* Dangerous aliasing;
* Modifications and global variable uses that are inconsistent with specified interfaces;
* Problematic control flow such as likely infinite loops, fall through cases or incomplete switches, and suspicious statements;
* Buffer overflow vulnerabilities;
* Dangerous macro implementations or invocations;
* Violations of customized naming conventions.

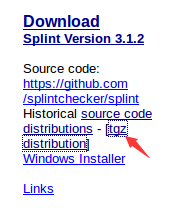
More details you can get from Splint User's Manual [link](http://www.splint.org/manual/manual.html#memory).

With such knowledge, your goal is to achieve the followings:

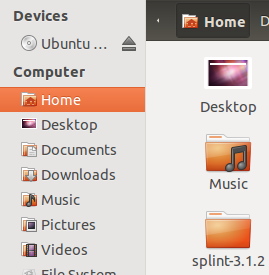
* Install splint;
* Finish code samples with 2 different kinds of problems which can be detected by Splint. You can choose any 2 of 11 problems as above.
* Use splint to detect the 2 kinds of problems. Descibe your observations in your report.

Steps :

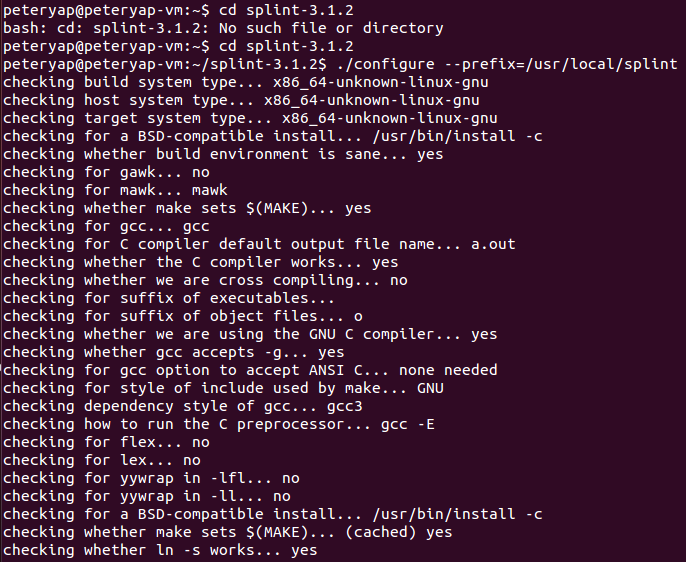
1. Download tgz distribution on the splint.org website.

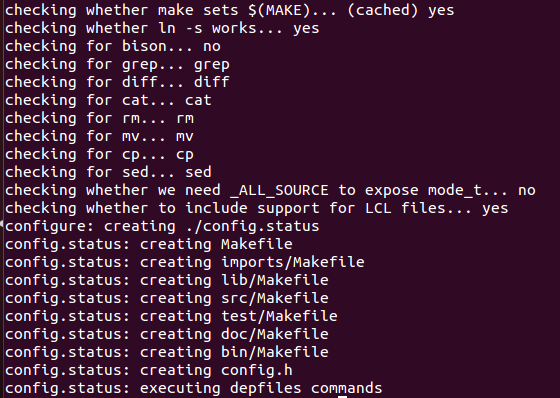


1. Extract the .tgz file and copy the splint-3.1.2 file to home page.

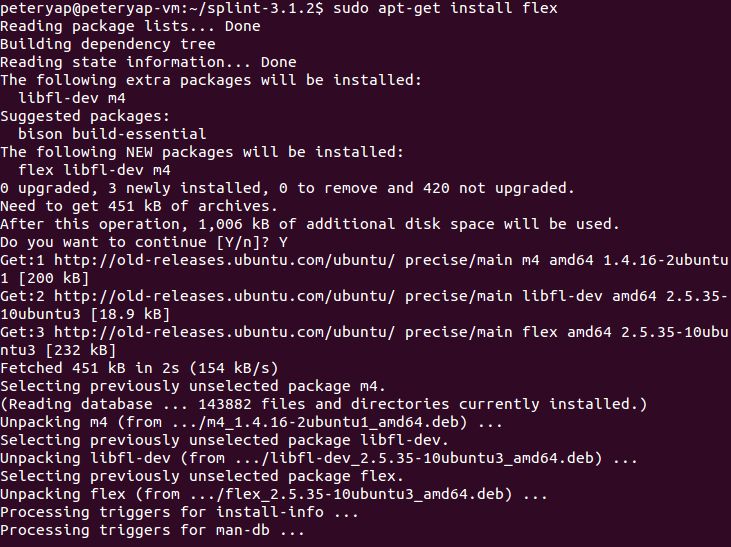


1. Setup Splint.





Command “sudo apt-get install flex”



Command “make” configure too much file, so there is no screenshots.

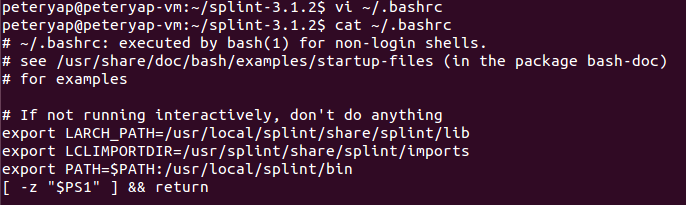
Command “sudo make install” same too.

1. Use vi Text Editor to add the following to the environment variable.

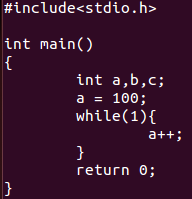
export LARCH\_PATH=/usr/local/splint/share/splint/lib

export LCLIMPORTDIR=/usr/splint/share/splint/imports

export PATH=$PATH:/usr/local/splint/bin

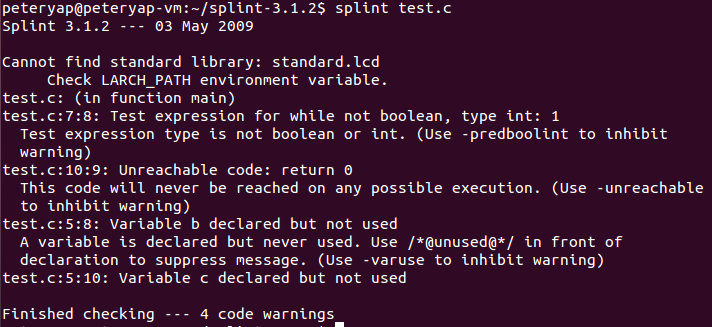


1. Execute source ~/.bashrc
2. Write a code example with two different kind of problems. “vi test.c”



Problems exist are dead loop and variable has not been used in program.

1. Use splint to detect the problems in the c program.



1. Splint warning shows that the while expression is not Boolean and exists in dead loop, variable is declared but never used in the program. The problems exist in the c program are found, then the lab is completed.

**Lab 3.2 Using eclipse for java static analysis**

**Overview**

The learning objective of this lab is for students to gain the first-hand experience on using static code analyzers in Eclipse to check Java program for security vulnerabilities and coding mistakes.

In this Lab, your goal is to achieve the followings:

* Install plugins in Java;
* Learn to check Java code by using static code analyzers in Eclipse. Descibe your observations in your report.

**Open Source Code Analyzers in Java**

Here we introduce 3 kinds of open source code analyzers in Java.

**FindBugs**

FindBugs looks for bugs in Java programs. It can detect a variety of common coding mistakes, including thread synchronization problems, misuse of API methods, etc. Go To FindBugs[link](http://findbugs.sourceforge.net/).

**PMD**

PMD scans Java source code and looks for potential problems like:

 \* Unused local variables   
 \* Empty catch blocks   
 \* Unused parameters   
 \* Empty 'if' statements   
 \* Duplicate import statements   
 \* Unused private methods   
 \* Classes which could be Singletons   
 \* Short/long variable and method names

Go To PMD [link](http://pmd.sourceforge.net/).

**Checkstyle**

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard. Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration file is supplied supporting the Sun Code Conventions. As well, other sample configuration files are supplied for other well known conventions. Can be integrated into CruiseControl and Eclipse. Go To Checkstyle [link](http://checkstyle.sourceforge.net/).

**Two Ways of Installing Eclipse Plugin**

Here we introduce two normal ways of installing Eclipse Plugins.

**Install plugins online**

All plugins can be installed with the standard update manager of Eclipse. Following is the procedure:

\* Choose the menu item "Help->Intall New Software".  
\* Click on "Add..." and enter a name and the URL of the plugin.  
\* Now you can select the feature (plugin) you wish to install.  
\* Confirm the upcoming dialogs and restart Eclipse.

**Install plugins offline**

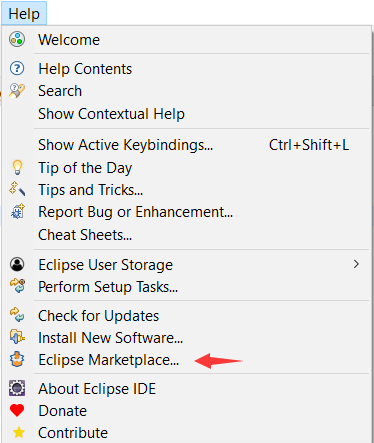
For Eclipse 3.4 or later, you can install downloaded plugins by copying the contents of plugins to the folder "dropins". And you can delete the plugins in folder "dropins" to uninstall plugins. After installing or uninstalling, you need to restart Eclipse.

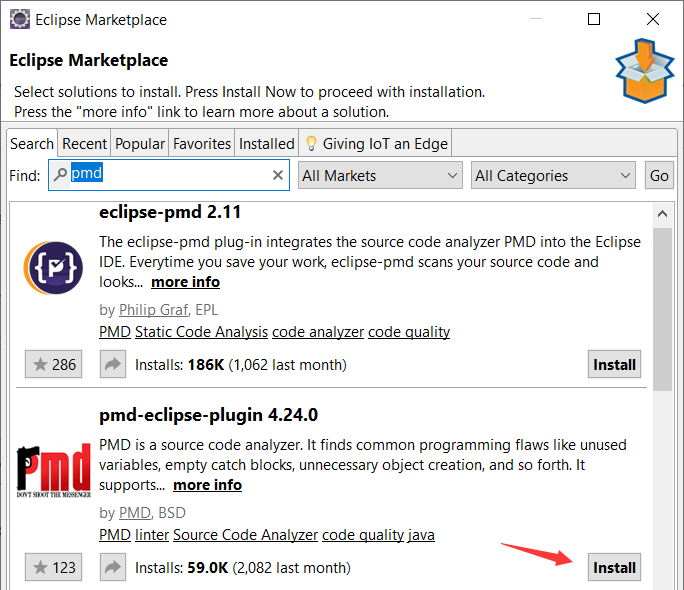
Steps :

1. Install PMD Plugins

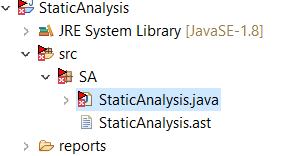
(Because the link <http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/> cannot be add to the repository, so we download PMD plugin in others way.)

Eclipse -> Help -> Eclipse MarketPlace -> Search “PMD” in the box -> Select the PMD-eclipse-plugin but not Eclipse-PMD. Click on install and restart eclipse then its done.

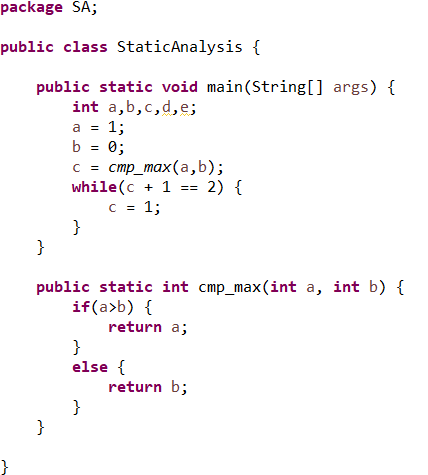




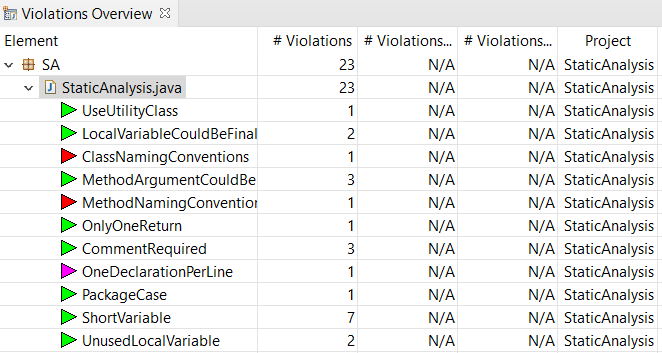
1. Create arbitrary sample Java project.

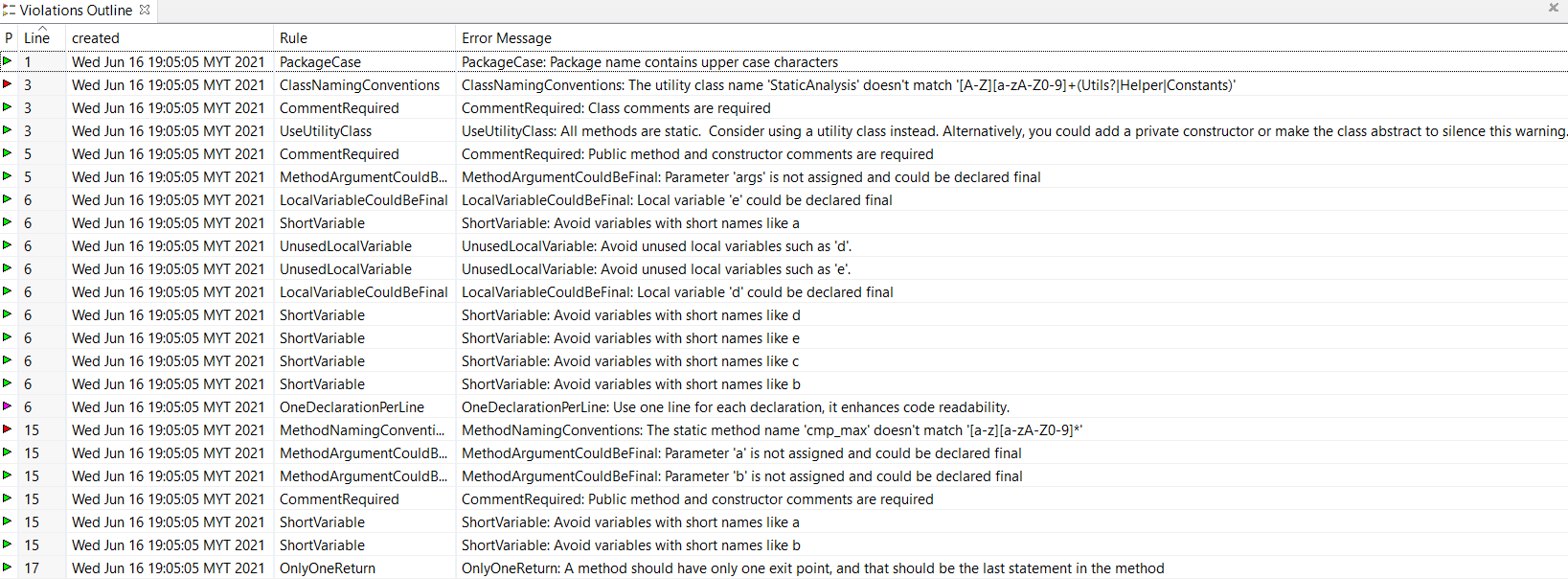


Java Code



Check Code Violation





There is a pmd-report text file showing the analysis.



1. PMD plugin help us to find those problems in the java program. We can click the violation and see whether which problem exists. This lab is completed.