

## 实验六：C/C++静态源码分析

### 一、实验目标

1. 理解静态程序分析在项目中的作用
2. 学会使用 cppcheck 等静态源码分析工具
3. 理解静态程序分析工具与编译器的不同作用
4. 理解静态程序分析工具会对那些错误进行检查

### 二、内容与要求

#### 1、安装

Ubuntu 环境下使用命令：

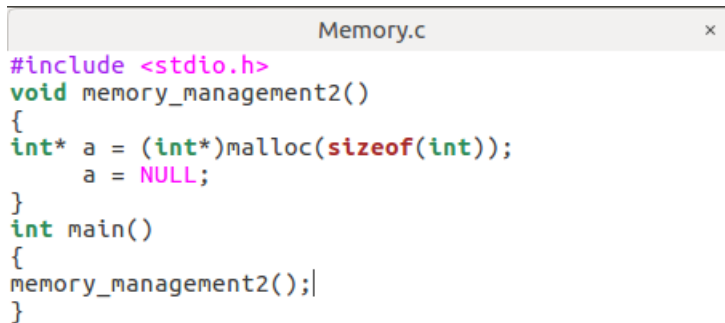
Sudo apt update

sudo apt-get install cppcheck

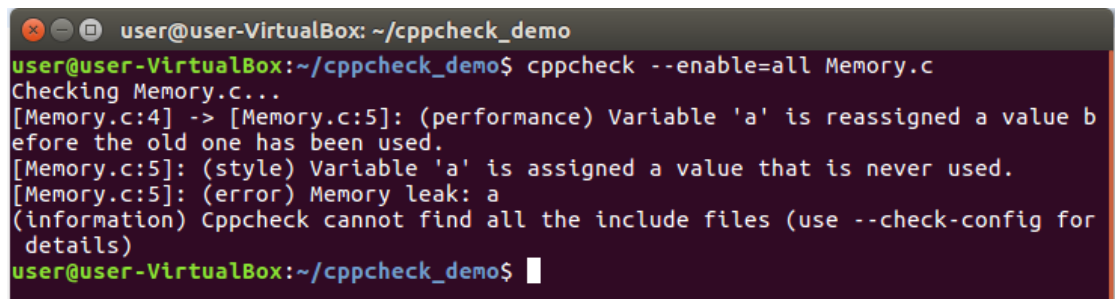
或在 github 下载最新版本：

#### 2、示例

cppcheck --enable=all Memory.c



```
Memory.c
#include <stdio.h>
void memory_management2()
{
    int* a = (int*)malloc(sizeof(int));
    a = NULL;
}
int main()
{
    memory_management2();
}
```



```
user@user-VirtualBox: ~/cppcheck_demo
user@user-VirtualBox:~/cppcheck_demo$ cppcheck --enable=all Memory.c
Checking Memory.c...
[Memory.c:4] -> [Memory.c:5]: (performance) Variable 'a' is reassigned a value before the old one has been used.
[Memory.c:5]: (style) Variable 'a' is assigned a value that is never used.
[Memory.c:5]: (error) Memory leak: a
(information) Cppcheck cannot find all the include files (use --check-config for details)
user@user-VirtualBox:~/cppcheck_demo$
```

#### 3、修改不报错

```
*Memory.c
#include <stdio.h>
void memory_management2()
{
    int* a = (int*)malloc(sizeof(int));
    free(a);
    a = NULL;
}
int main()
{
    memory_management2();
}
```

```
user@user-VirtualBox: ~/cppcheck_demo
user@user-VirtualBox:~/cppcheck_demo$ cppcheck --enable=all Memory.c
Checking Memory.c...
[Memory.c:4] -> [Memory.c:5]: (performance) Variable 'a' is reassigned a value before the old one has been used.
[Memory.c:5]: (style) Variable 'a' is assigned a value that is never used.
[Memory.c:5]: (error) Memory leak: a
(information) Cppcheck cannot find all the include files (use --check-config for details)
user@user-VirtualBox:~/cppcheck_demo$ cppcheck --enable=all Memory.c
Checking Memory.c...
(information) Cppcheck cannot find all the include files (use --check-config for details)
user@user-VirtualBox:~/cppcheck_demo$
```

### 三、Cppcheck 使用手册

\$cppcheck ./ (检查当前文件夹下所有文件)

\$cppcheck - -enable=all demo.c (--enable=all 启用所有检查)

(--enable=①warning②performance③information④style⑤all)

\$cppcheck -i demo.c [filepath] (-i 排除指定文件，检查指定路径下除了 demo.c 外的所有文件)

不确定消息: --inconclusive(即使分析不确定，也会报告)

保存分析结果:(\$cppcheck example.cpp 2>err.txt)

启用多线程: \$cppcheck -j 4 demo.c (4 个线程)