Cppcheck 软件用户使用手册

二零一四年十一月二十日

目录

1 引言

Cppcheck 是一个 C/C++代码分析工具，由 Daniel Marjamaki 与 Cppcheck 团队共同维护。Cppcheck 是目前开源项目中，最好的 C/C++代码分析工具。Cppcheck 的源代码可以在 GitHub 上获得。Cppcheck 计划每一、二个月会发布一个新的版本，截止至 2014 年 11 月 20 日，Cppcheck 的版本号为 1.67。

本使用手册，向用户介绍了 Cppcheck 的运行环境、软件功能、软件安装以及使用方法等。

手册说明

```
$ cd ./cppcheck/
```

上例中的美元符($) 代表命令行提示符，是用于提示用户需要在命令行中输入除美元符($) 外的其他内容(cd ./cppcheck/)。

英文缩写

GUI – Graphic User Interface – 可视化界面

CPU – Central Processing Unit – 中央处理器

PCRE – Perl Compatible Regular Expressions – Perl 兼容的正则表达式库

参考资料

Cppcheck 项目主页：http://cppcheck.sourceforge.net/

Cppcheck GitHub 项目地址：https://github.com/danmar/cppcheck/

2 简介

Cppcheck 是一个 C/C++代码分析工具，最开始的名称是”C++check”，而后改称为”Cppcheck”。

和 C/C++编译器、其他的许多分析工具不一样的是，Cppcheck 并不会检查语法错误。Cppcheck 旨在检查编译器一般检查不到的错误，它的目标之一是不误报。

另一方面，Cppcheck 有一定的局限性。之所以 Cppcheck 很少误报，是因为有许多错误 Cppcheck 并不检查。

仔细测试代码，你会找到更多的错误。不过当你没能找到一些错误时，Cppcheck 仍会是一个好工具。

3 运行环境

3.1 硬件要求

Cppcheck 对内存和 CPU 的要求不高，绝大多数计算机都可以运行 Cppcheck。

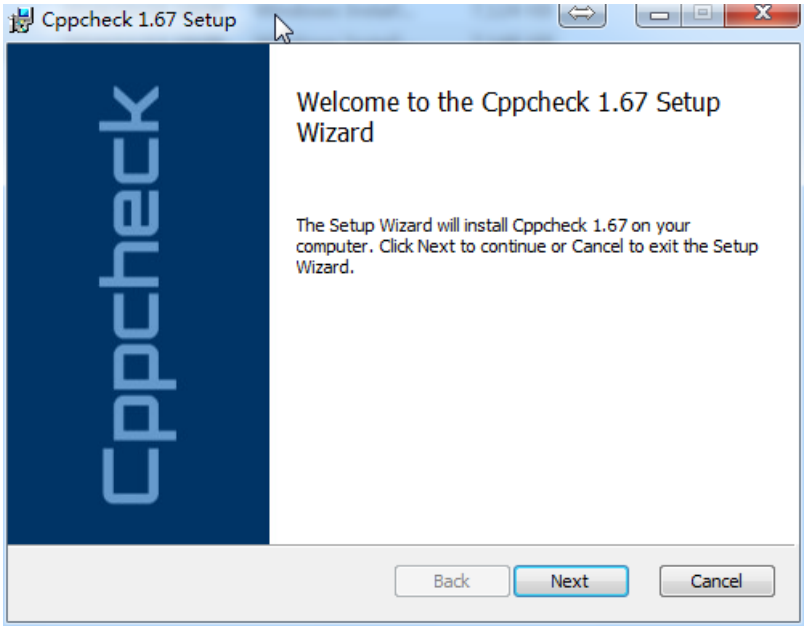
3.2 支持的操作系统

Cppcheck 支持 Windows 和 Linux。提供 Cppcheck Windows 安装文件，Linux 需要下载源码再编译。

4 安装

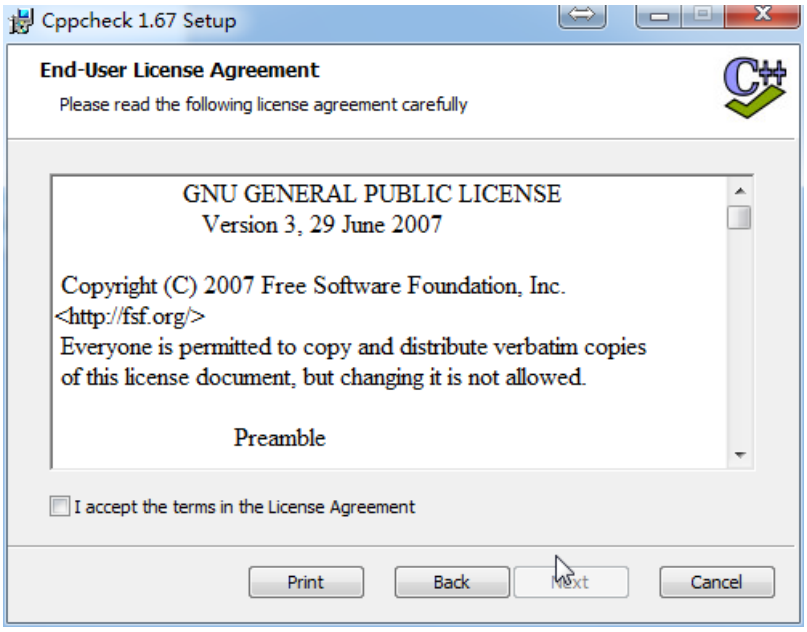
4.1 Windows

下载 Windows 版本的安装文件。运行 Cppcheck Windows 安装文件。如图一。



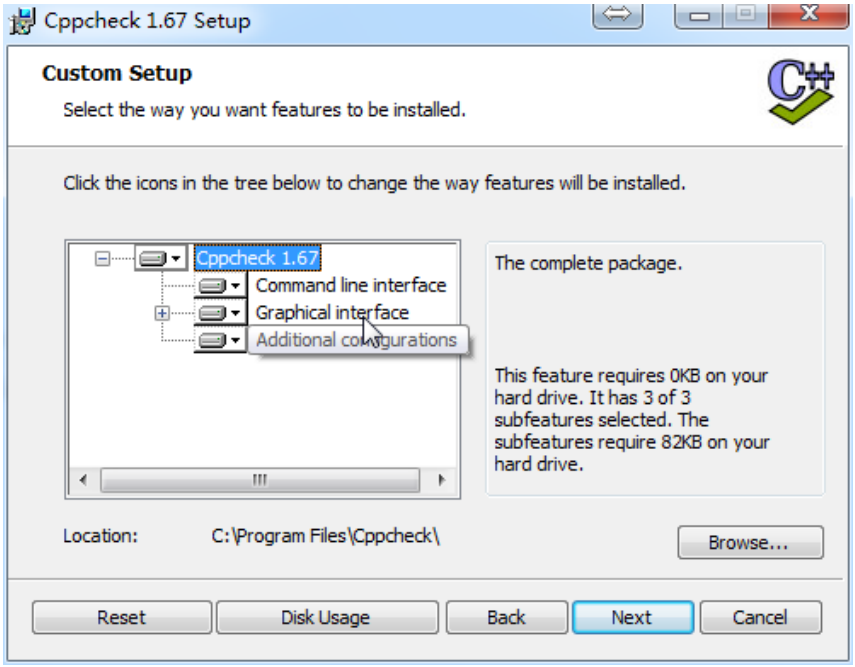
图一 安装界面（一）

点击 Next 按钮，进入版权声明页面。Cppcheck 采用的是 GNU License。如图二。



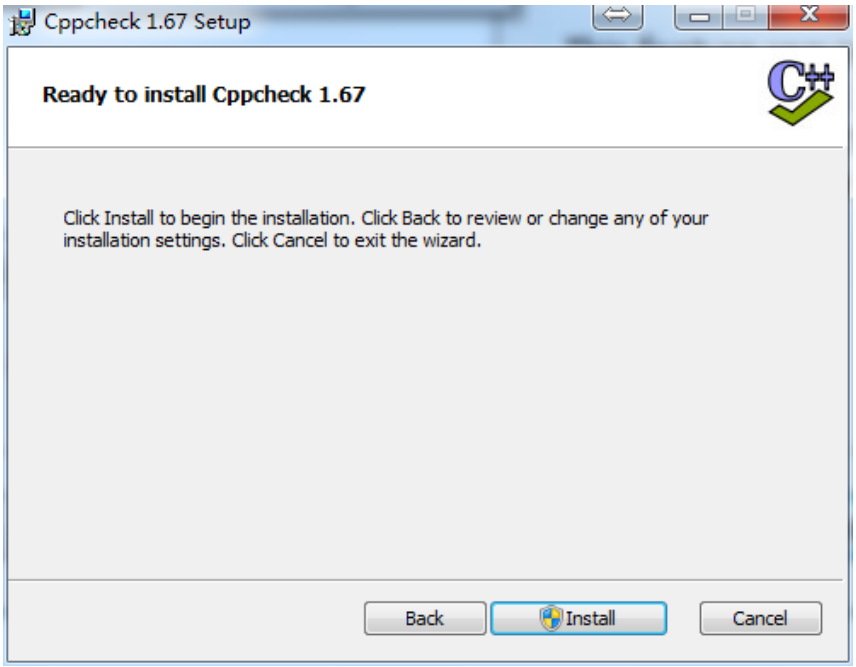
图二 安装界面（二）

勾选”I accept the terms in the License Agreement”选框，点击 Next 按钮进入功能选择及程序安装路径选择页面。功能分为三个部分，命令行形式、图形化界面和额外的配置。一般选择全部安装。如图三。



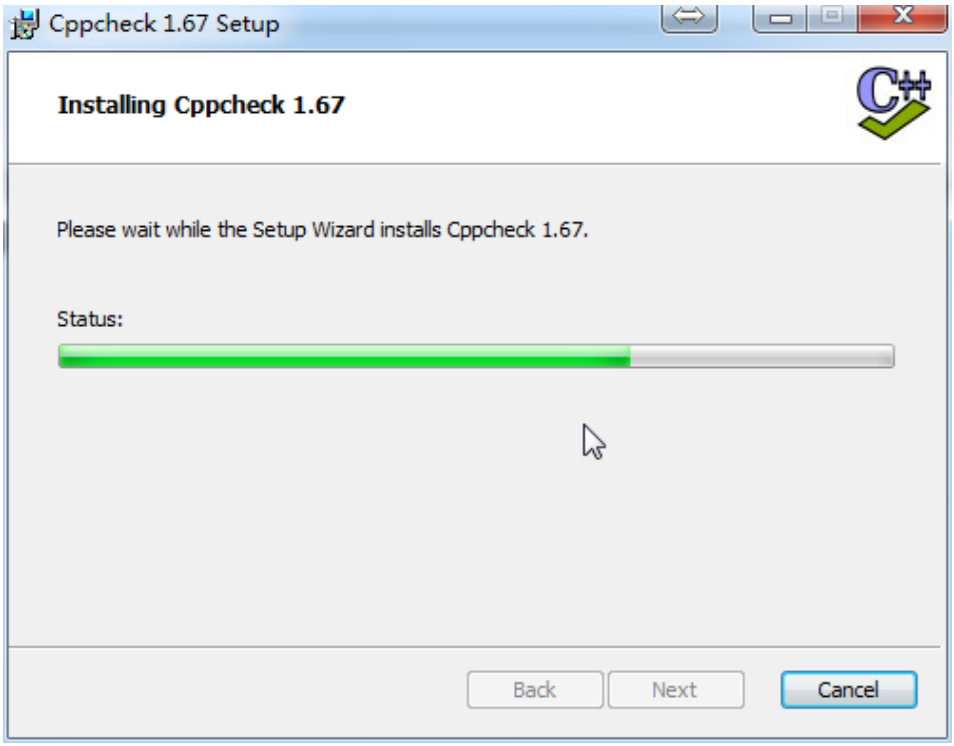
图三 安装界面（三）

确认好安装的功能及程序安装路径，点击 Next 按钮，进入确认安装界面。如图四。



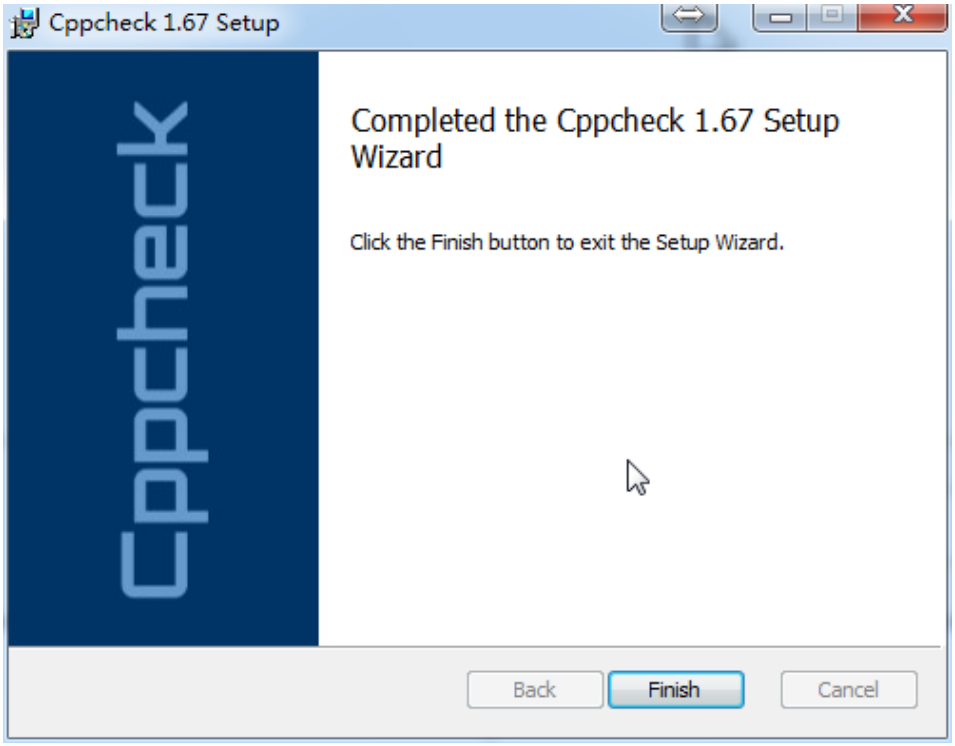
图四 安装界面（四）

点击 Next 按钮，开始执行安装操作。安装过程如图五所示。



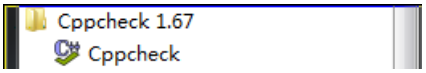
图五 安装界面（五）

程序安装好之后会有一个提示页面，如图六。

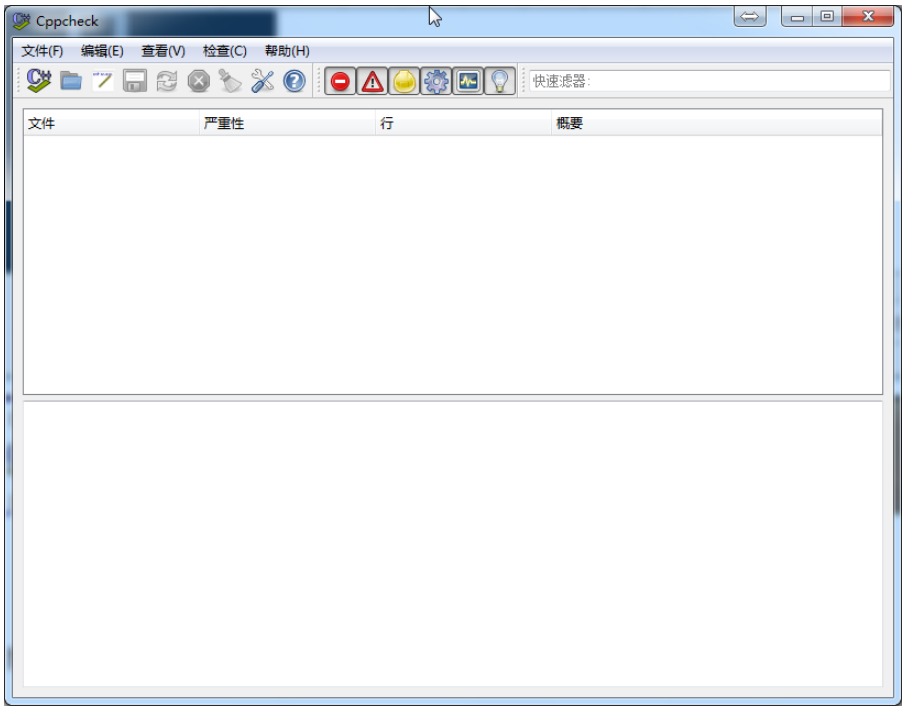


图六 安装完成界面

安装完成后，可以在开始菜单中找到 Cppcheck 程序。如下图所示。



运行 Cppcheck，你可以看到一个可视化界面（图八），Cppcheck 已经可以使用了。



图八 Cppcheck 的 Windows 可视化界面

点击帮助菜单下的关于，会弹出一个关于 Cppcheck 介绍的窗口，如图九所示。

图九 Cppcheck 介绍窗口

4.2 编译安装（Linux）

4.2.1 下载源码

Cppcheck 在 GitHub 上提供源代码下载。可以使用 git 工具：

```
$ git clone git://github.com/danmar/cppcheck.git
```

或使用 svn 工具：

```
$ svn checkout https://github.com/danmar/cppcheck/trunk
```

4.2.2 准备工作

下载好源代码后，可以使用各类最新的 C++编译器（支持或部分支持 C++11）进行编译。如果需要编译 GUI，那么需要安装 Qt 库。如果需要支持 Rules 功能，需要安装 PCRE 库。Qt 库和 PCRE 库是可选的，没有安装 Qt 库或 PCRE 库仍可以编译 Cppcheck 的主程序。

拿到代码后，决定好需要安装的功能并准备好依赖关系，我们可以选择各类编译方式。本手册将以 git 工具为例，介绍 GNU make 的方法。用户可以使用 Visual Studio 或是 Qt Creator 来编译 Cppcheck，具体方法可以参看源码中的 readme.txt 文件。

4.2.3 拿到最新稳定版

使用 git 下载到代码后，进入代码目录，检查代码最新版本，使用 git tag 命令查看：

```
$ git tag | tail -1
1.67
```

然后签出 1.67 版本的代码（使用 git checkout 命令）：

```
$ git checkout 1.67
```

4.2.4 编译代码

执行 make。（不使用额外的 PCRE 库支持）

```
$ cd ./cppcheck/
$ make
```

如果需要 PCRE 库支持，则需要在 make 后添加参数。

```
$ cd ./cppcheck/
$ make SRCDIR=build CFGDIR=cfg HAVE_RULES=yes
```

其中，SRCDIR=build 指定使用 Python 优化 Cppcheck，CFGDIR=cfg 指定.cfg 文件存储的目录，HAVE_RULES=yes 激活 Rules 功能（这功能需要 PCRE 库）。

4.2.5 确认无误

编译完成后会在目录下生成 cppcheck 可执行程序，运行 cppcheck --version 可以查看编译好的 cppcheck 的版本号。

```
$ ./cppcheck --version
Cppcheck 1.67
```

5 常用操作

以下我们介绍命令行运行 cppcheck 的方法以及 GUI 程序的操作方法。

5.1 第一个例子

假设我们有如下一段简单的代码：

```
$ cat file1.c
int main() {
    char a[10];
    a[10] = 0;
    return 0;
}
```

我们可以使用 cppcheck 来检查：

```
$ ./cppcheck file1.c
```

程序的输出会是：

```
[file1.c:4]: (error) Array 'a[10]' accessed at index 10, which is out of bounds.
```

5.2 检查一个目录

一般情况下，一个程序会有许多的源代码文件。我们需要检查所有的源代码。cppcheck 可以检查一个目录下所有的源代码。

```
$ ./cppcheck path
```

上例中的”path”是目录的时候，cppcheck 就会检查这个目录下，所有的源文件。

```
Checking path/file1.cpp...
1/2 files checked 50% done
Checking path/file2.cpp...
2/2 files checked 100% done
```

5.3 忽略指定的文件（目录）

使用 cppcheck 工具检查源代码时，忽略指定文件或目录的方法有两种。

一是只指定要检查的代码：

```
$ ./cppcheck src/a src/b
```

以上只会检查 src 目录下的 a 文件和 b 文件。

二是使用-i 参数来明确指出忽略的源文件：

```
$ ./cppcheck –isrc/c src
```

以上会检查 src 目录下除 c 文件外其他的文件。

5.4 报错消息的分级

cppcheck 报错消息有如下的等级：

等级	内容
error（错误）	指出找到的错误（bug）
warning（警告）	预防错误的建议
style(风格警告)	和代码整洁性（未用到的函数、重复的代码、不会发生变化的常量等等）有关的问题
performance（性能警告）	可以让代码更快一点的建议。这些建议都是基于很基础的知识，你不能指望这些建议给你带来显著的性能上的提升。
Portability（可移植性警告）	关于移植性的问题。如 64 位可移植性，代码在不同编译器下会有不一样的情形，等等。
Information（信息）	检查源程序过程中的问题。

5.5 设置消息显示的等级

默认情况下，只会显示 error（错误）。使用--enable 参数，我们可以让 cppcheck 显示更多的消息。

```
# 显示 warning（警告） 消息
$ ./cppcheck --enable=warning file.c

# 显示 performance（性能警告） 消息
$ ./cppcheck --enable=performance file.c

# 显示 information（信息） 消息
$ ./cppcheck --enable=information file.c

# 同时显示 performance（性能警告）、 portability（可移植性警告）和
# style（风格警告）
$ ./cppcheck --enable=style file.c

# 显示 warning（警告）和 information（信息）消息
$ ./cppcheck --enable=warning, information file.c

# 显示 unusedFunction（未用到的函数）消息
$ ./cppcheck --enable=unusedFunction file.c

# 显示 all（所有）的消息
$ ./cppcheck --enable=all file.c
```

需要注意的一点是，参数 --enable=unusedFunction 应该只在要检查整个程序的时候才用。同样的，参数 --enable=all 也应该只在要检查整个程序的时候才用。主要的原因是，未使用的函数会在这两种检查下报错，所以如果 cppcheck 每看到一个没有被调用的函数就报错，就会显得很烦人。

提示：默认情况下，cppcheck 只会显示它有把握的错误。你可以使用 --inconclusive 参数来让 cppcheck 显示那些它没把握的错误。

```
$ ./cppcheck --inconclusive file.c
```

这可能会引起误报，cppcheck 可能会找到一些并不是错误的错误。请在使用前，确认可以接受误报。

5.6 保存结果

当你想要保存结果的时候，你可以使用 Linux 的 bash 自带的管道流（pip）的重定向：

```
$ ./cppcheck file.c 2> err.txt
```

5.7 多线程

参数-j 可以用来指定线程数量。比如，想要同时使用四个线程检查一个目录：

```
$ ./cppcheck -j 4 path
```

6 输出 XML

cppcheck 可以输出 XML 格式的结果。XML 格式有两个版本的格式，版本一（Version 1）和版本二（Version 2）。旧版本（版本一）的存在，仅仅是为了和以前的 CppCheck 兼容，也不会发生变化了。但是在将来，旧版本（版本一）很有可能会被移去，所以请尽量使用新版本（版本二）的 XML 输出。

使用重定向流的方法输出 XML 到指定的文件。下面的例子把 XML 报告输出到 out.txt 文件内。

使用旧版本命令是：

```
$ ./cppcheck --xml file1.cpp 2> out.txt
```

使用新版本命令是：

```
$ ./cppcheck --xml-version=2 file1.cpp 2> out.txt
```

下面是一份新版本 xml 报告的样本：

```
<?xml version="1.0" encoding="UTF-8"?>
<results version="2">
    <cppcheck version="1.67"/>
    <errors>
        <error id="someError" severity="error" msg="short error text"
verbose="long error text" inconclusive="true">
            <location file="file.c" line="1"/>
        </error>
    </errors>
</results>
```

6.1 <error>元素

cppcheck 把检查到的每个错误，都放在一个<error>元素中。<error>元素有如下的属性：

属性	说明
id	消息的 id。一些由符号组成的名称。
severity	Error, warning, style, performance, portability 和 information 之一。
msg	消息具体内容的简短描述。
verbose	消息具体内容的详细描述。
inconclusive	这个属性只会在允许 cppcheck 误报时，出现在那些不确定的消息里。

6.2 <location>元素

所有有关某个错误的位置信息，分别被放在一个<location>元素中。最主要的位置信息会被放在第一个<location>元素中。<location>元素有如下的属性：

属性	说明
file	文件名。可能是相对路径，也可能是绝对路径。
line	一个行号。

7 自定义输出格式

如果你想要改变输出格式，可以使用模板。

如果想要得到类似于 Visual Studio 的输出，可以使用--template=vs 参数：

```
$ ./cppcheck --template=vs gui/test.cpp
Checking gui/test.cpp...
gui/test.cpp(31): error: Memory leak: b
gui/test.cpp(16): error: Mismatching allocation and deallocation: k
```

如果想要得到类似 GCC 的输出，可以使用--template=gcc 参数：

```
$ ./cppcheck --template=gcc gui/test.cpp
Checking gui/test.cpp...
gui/test.cpp:31: error: Memory leak: b
gui/test.cpp:16: error: Mismatching allocation and deallocation: k
```

你也可以定制自己的格式（下面是一个以逗号为分隔符的例子）：

```
$ ./cppcheck --template="{file},{line},{severity},{id},{message}" gui/test.cpp
Checking gui/test.cpp...
gui/test.cpp,31,error,memleak,Memory leak: b
gui/test.cpp,16,error,mismatchAllocDealloc,Mismatching allocation and
deallocation: k
```

以下是支持的格式符：

格式符	说明
file	文件名
id	消息 id
line	行号
message	详细的信息
severity	消息等级

也可以在格式中使用转义序列如\b，\n，\r，\t 等。

9 生成 HTML 报告

cppcheck 提供了把 XML 输出转为 HTML 报告的方法。该方法需要先安装 Python 和 pygments 模块（可以从 http://pygments.org 获取）。在 cppcheck 源代码目录中，有一个目录是 htmlreport，里面有一个将 cppcheck XML 输出文件转为 HTML 输出文件的脚本。

可以使用如下命令查看帮助：

```
$ htmlreport/cppcheck-htmlreport -h
```

会在终端界面生成如下的帮助：

```
Usage: cppcheck-htmlreport [options]
Options:
  -h, --help            show this help message and exit
  --file=FILE           The cppcheck xml output file to read defects from.
                        Default is reading from stdin.
  --report-dir=REPORT_DIR
                        The directory where the html report content is written.
  --source-dir=SOURCE_DIR
                        Base directory where source code files can be found.
  --source-encoding=SOURCE_ENCODING
                        Encoding of source code.
```

下面是一个用例：

```
$ ./cppcheck gui/test.cpp -xml 2>err.xml
$ htmlreport/cppcheck-htmlreport --file=err.xml --report-dir=test1 --source-dir=.
```

10 可视化界面

10.1 简介

cppcheck 提供一个可视化界面程序。

当可视化界面程序启动后，我们就可以看到可视化界面。

10.2 检查源代码

使用 Check（检查）菜单。

10.3 查看结果

结果以列表的形式被展示出来。

在 View（视图）菜单，可以选择显示/隐藏指定的消息。

可以将结果保存为 XML 文件。”文件”→“保存结果到文件”

10.4 设置

可以在 Language（语言）菜单下选择语言。

在编辑（Edit）→偏好（Preference）中有更多设置选项。