# 基于持续集成的 PC-Lint 静态检查

姜 文,刘立康
(西安电子科技大学 通信工程学院,陕西 西安 710071)

**摘 要**:PC-Lint 是一款历史悠久、使用广泛的静态代码检查工具。为了保证 C/C++软件产品的质量,许多软件开发组织都把 PC-Lint 检查作为代码走查的第一道工序。PC-Lint 工具提供了大量的检查选项,需要合理地配置检查选项,从而减少误报,提高查错、排错的效率。通过四个常用类型的应用实例,详细叙述了 PC-Lint 工具在各种应用和开发环境中 lnt 文件配置方法。详细叙述了基于持续集成 PC-Lint 的静态检查方法,以 ClearCase 作为软件配置管理工具,将 PC-Lint 静态检查工具集成到持续集成工具 ICP-CI 上,对 C/C++软件模块进行静态检查,有助于发现更多的源代码缺陷。最后介绍了 PC-Lint 静态检查的应用案例。长期的工作实践表明,进行 PC-Lint 静态检查有助于及时发现 C/C++软件代码的各种缺陷,从而提高软件代码的质量和降低软件开发的成本。

**关键词**:PC-Lint;静态检查;lnt 配置文件;持续集成
**中图分类号**:TP311.56 **文献标识码**:A **文章编号**:1673-629X(2016)11-0031-06
doi:10.3969/j.issn.1673-629X.2016.11.007

# PC-Lint Static Checking Based on Continuous Integration

JIANG Wen, LIU Li-kang
(School of Telecommunication Engineering, Xidian University, Xi'an 710071, China)

**Abstract**: PC-Lint is a static checking tool which has long history and wide application. In order to guarantee the quality of the code of C/C++, many software development organizations choose the inspection of PC-Lint as the first process during the code review. Plenty of inspectional options are provided by PC-Lint tool, and the inspectional options need to be deployed reasonablely, thus reducing the misinformation, increasing the efficiency of debugging and misarrangement. Through four common types of application examples, it describes method of configuration of lnt files in all kinds of application and development of environment in detail, and discusses the approach to static checking of PC-Lint based on continuous integration. Using ClearCase as the software configuration management tool, the static checking tool PC-Lint is integrated into the continuous integration tool ICP-CI, which contributes to find more source code defects by doing static checking for the C/C++ modules. At last, application case for PC-Lint static checking is introduced. Practice of long time shows that the PC-Lint static checking has conducted to discover all kinds of defects of the code in C/C++ timely, to improve software quality and reduce the cost of software development.

**Key words**: PC-Lint; static checking; lnt configuration file; continuous integration

## 0 引 言

PC-Lint[1-7]是一款历史悠久、功能异常强劲的静态代码检测工具。它的使用历史可以追溯到计算机编程的早期(30 多年以前)。经过多年的发展,PC-Lint 在全球拥有广泛的客户群,许多大型的软件开发组织都把 PC-Lint 检查作为代码走查的第一道工序。PC-Lint 小巧方便,使用广泛,是一款很有价值、可以随身携带的 C/C++语言代码静态检查工具。

PC-Lint 工具提供了大量的检查选项,合理地配置检查选项才能提高检查效率。文中通过四个常用类型的应用实例,叙述了 PC-Lint 工具在各种应用和开发环境中的 lnt 文件配置方法。介绍了基于持续集成 PC-Lint 静态检查方法以及 PC-Lint 静态检查的应用案例。工作实践表明,进行 PC-Lint 静态检查有助于及时发现 C/C++代码的各种缺陷,从而提高软件代码的质量,降低软件开发的成本。

**收稿日期**:2016-01-29 **修回日期**:2016-05-18 **网络出版时间**:2016-10-24
**基金项目**:国家部委基础科研计划:国防预研基金项目(A1120110007)
**作者简介**:姜 文(1986-),女,工程师,硕士研究生,CCF 会员,研究方向为图像处理与分析、数据库应用和软件工程;刘立康,副教授,研究方向为数字通信、图像传输与处理、图像分析与识别等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20161024.1117.078.html

## 1 PC-Lint 工具介绍

PC-Lint 是 GIMPEL SOFTWARE 公司开发的 C/C++软件代码静态分析工具,它的全称是 PC-Lint/FlexeLint for C/C++,目前的最新版本是 9.0.0L。

### 1.1 PC-Lint 的特点

(1)PC-Lint 是一种更加严格的编译器,不仅可以像普通编译器那样检查出一般的语法错误,还可以检查出那些虽然完全合乎语法要求,但很可能是潜在的、不易发现的错误。

(2)PC-Lint 可以检测单个文件,也可以同时检查多个文件,检查当前文件的同时还会检查所有与之相关的文件。通常也用于检查比较大的软件模块。

(3)支持 Scott Meyers[8-9]的名著(《Effective C++》和《More Effective C++》)中描述的各种提高效率和防止错误的方法。

(4)PC-Lint 支持几乎所有流行的编译器,支持大多数主流嵌入式系统的编译器,拥有很多支持异类编译器的选项。

(5)PC-Lint 支持几乎所有流行的编辑环境,比如 Visual Studio、Source insight 等。

(6)语言兼容性好,支持 K&R C、ANSI、ANSI/ISO C++。

### 1.2 PC-Lint 告警消息分类

PC-Lint 能够检查出很多语法错误和语法上正确的逻辑错误,为大部分错误消息都分配一个错误编号。PC-Lint 告警消息的详细分类如表 1 所示。

表 1 PC-Lint 告警消息分类

| 错误说明 | C | C++ | 告警级别 |
|---|---|---|---|
| 语法错误 | 1 ~ 199 | 1 001 ~ 1 199 | 1 |
| 内部错误 | 200 ~ 299 | | 0 |
| 致命错误 | 300 ~ 399 | | 0 |
| 告警 | 400 ~ 699 | 1 400 ~ 1 699 | 2 |
| 消息 | 700 ~ 800 | 1 700 ~ 1 899 | 3 |
| 可选信息 | 900 ~ 999 | 1 900 ~ 1 999 | 4 |

其中,0 ~ 2 级告警都不能使用-e 选项进行屏蔽,只能使用-esym,-emacro,-sem 之类的选项进行屏蔽;3 级告警要根据具体情况来确认是否使用-e 选项进行屏蔽。

### 1.3 PC-Lint 工具的文件组成

PC-Lint 工具主要由以下文件组成,如表 2 所示。

其中,std.lnt,options.lnt 由用户自己编写,其他文件由 PC-Lint 工具提供。

### 1.4 PC-Lint 的代码检查功能

PC-Lint 的检查分很多种类,有强类型检查、变量值跟踪、语义信息、赋值顺序检查、弱定义检查、格式检查、缩进检查、const 变量检查和 volatile 变量检查等等。对于每种检查类型,PC-Lint 都有详细的检查选项,用以控制 PC-Lint 的检查效果。

表 2 PC-Lint 工具的文件组成与功能介绍

| 文件类型 | 文件名 | 功能 |
|---|---|---|
| 执行文件 | LINT-NT.exe | PC-Lint 执行文件 |
| | config.exe | PC-Lint 配置程序 |
| 文档 | PC-Lint.pdf | PC-Lint 帮助手册 |
| | msg.txt | 错误信息列表,根据它来修改代码 |
| 配置文件 | co.lnt | 通用编译器的配置选项 |
| | co-xxx.lnt | 特定编译器的配置选项 |
| | env-xxx.lnt | 特定编辑环境的配置文件 |
| | lib-xxx.lnt | 特定库的配置文件 |
| | au-xxx.lnt | 作者推荐检查项的配置文件 |
| | std.lnt | 标准配置文件,用户自己配置 |
| | options.lnt | 检查规则选项配置文件,用户自己编写 |

PC-Lint 可以发现 C/C++源代码的以下问题:变量声明了但未使用、变量类型不匹配、变量在使用前未定义、不可达代码、死循环、数组越界、内存泄漏等代码缺陷;能够对程式进行全局分析,识别没有被适当检验的数组下标,报告未被初始化的变量,警告使用空指针连同冗余的代码。它不但能够监测出许多语法逻辑上的隐患,而且也能够有效地提出许多 C/C++代码在空间利用、运行效率上的改进点。

## 2 PC-Lint 静态检查的 lnt 配置文件

PC-Lint 是一个命令行工具,提供了大概 300 多个选项,近 2 000 个告警,合理配置选项减少误报之后,可以大幅提高查错、排错的效率。开发人员需要根据软件模块自身的特点和环境编写一套 lnt 配置文件才能正确有效地使用 PC-Lint 工具。

### 2.1 lnt 配置文件内容和使用方法

#### 2.1.1 编写 lnt 配置文件

对于软件模块的 PC-Lint 检查,为了使用和修改方便,通常可以编写四个 lnt 配置文件,配置文件命名为 * _模块名.lnt。

(1)编译配置文件 co_模块名.lnt。

该文件为 PC-Lint 检查提供编译环境,通常有两种方式:

①选择 PC-Lint 工具的 lnt 文件提供编译环境,为了方便用户使用,PC-Lint 提供四类 lnt 文件供用户选用;

②采用宏定义的方式提供编译环境。

(2)检查路径配置文件 include_模块名.lnt。

该文件包括系统头文件路径、软件模块头文件路径。

(3)检查规则配置文件 options_模块名.lnt。

该类文件内容比较繁杂,常用的检查规则选项主要包括如下内容。

①检查级别统一控制选项,如-w3。

②库头文件检查选项,如-wlib(0)。

③告警信息条目的选择,根据需要增删其中的选项。

④-esym,-emacro 屏蔽特定告警选项,-function,-sem 增加某些检查选项。

⑤执行编译时需要的输出选项,检查和诊断时输出格式。

⑥其他选项:包括变量类型的缺省定义(如-si4,-sp4),定义保留字等。

在实际应用中该文件只包含上述部分内容,根据软件模块自身特点和静态检查的要求检查规则选项可多可少,有时可以为空文件。

(4)顶层配置文件 std_模块名.lnt。

该文件包含前面的三个文件:co_模块名.lnt、include_模块名.lnt、options_模块名.lnt。

2.1.2　PC-Lint 工具的使用方法

命令行的使用方式是 PC-Lint 最基本的使用方式,也是其他各种集成使用方式的基础。PC-Lint 的命令行有下列形式:

LINT-NT option file1 [file1 file3 …]

其中,LINT-NT 为 PC-Lint 在 Windows 平台上的可执行程序 LINT-NT.exe;option 为 PC-Lint 的配置文件;file 为待检查的源文件。

option 通常为顶层配置文件 std_模块名.lnt。

### 2.2　lnt 配置文件的应用实例

下面介绍几个 lnt 配置文件的编写实例。由于检查规则配置文件 options_模块名.lnt 没有固定的模式,在不同的应用中差别很大,在实例中不做详细介绍。顶层配置文件与 2.1.1 节中相同,实例中不再重复。

2.2.1　基于 Visual Studio IDE 的软件模块 lnt 文件配置

开发环境的操作系统为 Windows 7,采用 Visual Studio 10[10] 作为 C/C++语言的集成开发环境,以 olt 工具模块为例叙述 lnt 配置文件的内容。

(1)co_olt.lnt

```
au-sm123.lnt
au-ds.lnt
co-msc90.lnt
env-vc10.lnt
lib-atl.lnt lib-mfc.lnt
lib-stl.lnt lib-w32.lnt
```

(2)include.olt.lnt

```
//系统头文件路径
-IC:\Program Files\Microsoft Visual Studio 10.0\VC\include
-IC:\Program Files\Microsoft Visual Studio 10.0\VC\atlmfc\include
-IC:\Program Files\Microsoft SDKs\Windows\v7.0A\Include
//软件模块头文件路径
-ID:\Code_View\Olt_Tool
```

(3)options_olt.lnt

olt 模块检查规则选项,该文件可以为空文件。

可以采用命令行方式进行静态检查,也可以把 PC-Lint 工具集成到 Visual Studio 10 上检查代码。

2.2.2　基于 Linux 操作系统的软件模块 lnt 文件配置

编程环境的操作系统的版本是 64 位 Suse Linux 11.3,开发工具是 GNU Tools,使用 GCC 作为 C 代码的编译器,在 Windows 操作系统下进行软件模块 PC-Lint 静态检查。在 Windows 下用 PC-Lint 去分析 Linux C/C++代码是可以的,需要把 Linux 系统[11] 中/usr/下的一系列文件直接 COPY 到 Windows 系统中,包括系统定义的头(.h)文件和软件模块自己定义的头(.h)文件,从而保证可以正常使用 GCC 作为 C/C++代码的编译器。以 lop 模块为例叙述 lnt 配置文件的内容。在实例中采用宏定义的方式提供编译环境,也可以采用 PC-Lint 工具的 lnt 文件方式提供编译环境。

(1)co_lop.lnt

```
/*采用宏定义的方式提供编译环境,包括硬件、操作系统、编译器等内容*/
-DVOS_DISPATCHING_MODE=3
-DPFM_SYSTIME
-DGCC_VERSION
-D_GNU_SOURCE
-DVOS_HARDWRAE_PLATFORM=0
-DVOS_CPU_TYPE=0

-DVOS_OS_VER=4
-DLITTLE_ENDIAN
-D_CNUP_SUSELINUX
-D_CN_BYTE_ORDER=_CN_LITTLE_ENDIAN
-DVOS_BYTE_ORDER=VOS_LITTLE_ENDIAN
-DDEV_HOST_BYTE_ORDER=VOS_LITTLE_ENDIAN
-DDEV_NET_BYTE_ORDER=VOS_LITTLE_ENDIAN
-DVOD_FILE_SYSTEM_PROC

-DCAP_EFFECTIVE=0XF362F42E
-DCAP_PERMITED=0XF362F42E
```

(2)include_lop.ln

```
//系统头文件路径
-ID:\INC\WIN32
-ID:\INC\LINUX
-ID:\INC\LINUX\linux
//软件模块头文件路径
-I% PROJECT_PATH% \host\lop
```

-I% PROJECT_PATH% \public

2.2.3 基于 VxWorks 操作系统的软件模块 lnt 文件配置

VxWorks[12-13]是实时嵌入式操作系统软件,Tornado 是开发 VxWorks 应用系统的集成开发环境。Tornado IDE 采用 C/C++语言编程,支持 GNU C/C++编译器和 Diab C/C++编译器。可以在 Windows 操作系统下进行软件模块 PC-Lint 静态检查。以 sk 模块为例叙述 lnt 配置文件的内容。在实例中采用 PC-Lint 工具的 lnt 文件方式提供编译环境,也可以采用宏定义的方式提供编译环境。

(1) co. sk. lnt

au-sm. lnt au-ds. lnt

co. lnt //通用编译器

//co-gnu3. lnt

lib_at1. lntlib_corb. lnt lib-stl. lnt

lib_w32. lnt lib_wnt. lnt

(2) include. sk. lnt

//系统头文件路径

-I C\Tornado\target\h // VxWorks 头文件

-I C\Tornado\host\diab\include\cpp // iostreams 类库头文件

-I C\Tornado\host\x86-win32\i386-pc-mingw32\sys-Include

//软件模块头文件路径

-I % CODE_ROOT% \SK

-I % CODE_ROOT% \software

(3) options_sk. lnt

//sk 模块检查规则选项,该文件可以为空文件。

可以把 PC-Lint 工具集成到 Source Insight 编辑器(在编译配置文件中加入 env_si. lnt)上进行静态检查。许多大公司都在 VC++环境中实现了仿真 VxWorks 接口的代码(VxWorks 任务用 Windows 线程来仿真),可以在 VC++环境中编写和调试代码,适当调整编译配置文件 co. sk. lnt,也可以把 PC-Lint 工具集成到 VC 编辑器上进行静态检查。

2.2.4 软件产品的 lnt 文件配置

通常软件产品中包含的模块比较多,软件项目组首先编写一个各模块共用的 lnt 配置文件,该文件不允许软件模块开发人员修改,该文件内容包括编译环境的定义和检查规则选项。其他的 lnt 配置文件由开发人员根据模块自身的特点和环境自主编写。

## 3 基于持续集成的 PC-Lint 静态检查

文中采用的软件配置管理工具是 ClearCase,持续集成工具是 ICP-CI。通过自动化构建持续提供代码静态检查结果



### 3.1 PC-Lint 静态检查系统模型

持续集成[14]工具可以提供方便的集成平台,可以设置定时任务,使 PC-Lint 检查任务能够充分利用非工作时间完成比较耗时的静态检查过程,保证及时得到分析结果,方便开发人员根据检查结果报告修改软件代码缺陷。图 1 显示了一个基本的静态检查系统视图。

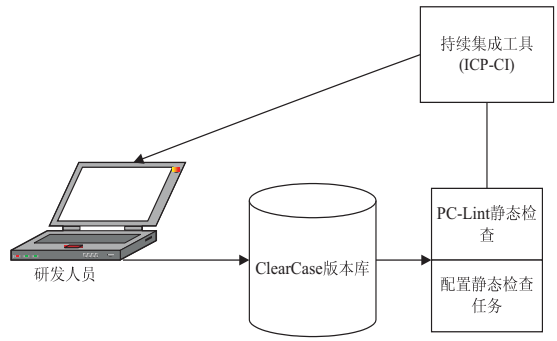


图 1 静态检查系统视图

构建流程如下:

(1)研发人员编写软件代码,对自己编写的软件代码进行 PC-Lint 静态检查。

(2)研发人员向软件配置管理(SCM)系统的版本库提交验证通过的源代码。

(3)持续集成工程师编写 *. lnt 配置文件、批处理文件、Makefile 文件和 ant 脚本。

(4)持续集成工程师在持续集成工具的任务管理页面上配置软件模块静态检查任务。

(5)持续集成工程师设定任务时间(通常是非工作时间),自动执行 PC-Lint 检查任务。

(6)持续集成工具通过电子邮件向软件产品经理和研发人员及时反馈静态检查结果。

(7)研发人员处理存在的各种问题。

基于持续集成的 PC-Lint 静态检查是对软件模块进行静态检查,通常一个软件模块包含几十个甚至上百个 C 语言源码文件。对模块进行静态检查有助于发现更多的源代码缺陷。

通常 PC-Lint 工具位于主控服务器与代理服务器的 plugin 目录下。持续集成工程师完成软件产品的 *. lnt 文件的配置之后,再搭建构建工程。构建工程命名为:“产品名_版本号_Pclint”。

### 3.2 ClearCase 版本库的代码更新

持续集成工具 ICP-CI 需要在版本库锁库之后完成源代码更新,然后进行 PC-Lint 检查。ICP-CI 工具执行代码更新时,需要编写代码更新的批处理脚本 code_update. bat,把代码更新的脚本配置在任务中。更新代码的批处理脚本内容如下:

VIEWPATH = D: \ClearCaseLOP\Code_View

Cleartool update-force-overwrtite-log "ccview. log" % VIEW-PATH%

### 3.3 ICP-CI 的任务管理页面上配置 PC-Lint 检查任务

在 ICP-CI 的任务管理页面上配置 PC-Lint 检查任务,通常由持续集成工程师来完成。配置时以模块为单位完成,以软件产品模块 lop 为例来描述集成过程。配置 lop 模块的 PC-Lint 任务时,在任务栏上选择"PC-Lint"任务。具体配置如表 3 所示。

表 3　lop 模块在 ICP-CI 页面上配置 PC-Lint 检查

| 参数名称 | 参数值 | 描述 |
|---|---|---|
| layerpath | /lop | 报告分层显示的层次名称,配置格式:"/模块名称" |
| workingdir | ${Code_Path}\host\lop | 设置 lnt 检查文件中引入的头文件的基目录 |
| sourcesPathRefID | pclint. path. id. lop | 设置 PC-Lint 检查的模块的代码路径 ID |
| lnt | ${dir. plugins}\pclint\lntfile\std_lop. lnt | 设置模块的顶层配置 lnt 文件 |
| runDistributed | false | 设置是否运行分布式 PC-Lint(默认为 flase) |
| xmlEncoding | UTF-8 | 设置 PC-Lint 生成 xml 报告的编码形式(默认为 UTF-8) |

配置各模块 PC-Lint 检查任务的 workingdir 的参数值时,可以在 ICP-CI 构建工程的"参数管理"页面添加"CODE_PATH"参数,并赋值为:D:\CODE_PATH。

配置各模块 PC-Lint 检查任务的 lnt 参数值时,${dir. plugins}是 ICP-CI 工具默认的各种检查工具放置路径。对于单机式系统,该路径为 ICP_CI_Windows_Master\plugins;对于主控式系统,该路径为 ICP_CI_Windows_Agent\plugins。

配置各模块 PC-Lint 检查的代码路径 ID 的参数值时,需要在 ICP_CI_Windows_Master\master\usrData\gourp\工程名\baseconfig. xml 文件中完成如下配置:

```
<path id="pclint. path. id. lop">
<fileset dir="${Code_Path}">
<includename="host\lop\*. cpp"/>
<includename="host\pub\*. c"/>
<includename="public\*. c"/>
</fileset>
</path>
```

完成以上配置后,在 ICP-CI 的 PC-Lint 任务配置页面上,对"sourcesPathRefID"进行配置时就可以选择新增的 ID 值"pclint. path. id. lop"。

配置各模块 PC-Lint 检查使用分布式部署时,需要将 runDistribute 的取值设置为 true 之后,再安装 IncrediBuild 工具来提升 PC-Lint 检查的速度。

### 3.4 PC-Lint 静态检查任务执行过程

ICP-CI 页面上配置 PC-Lint 任务后,会在 ICP_CI_Windows_Master\master\usrData\gourp\工程名下生成 pclint 任务的相关文件。ICP-CI 工具运行 pclint 任务时,运行过程如下:

(1)根据在 ICP-CI 页面上选择的"配置任务类型"自动引入以下 2 个文件:

```
<import file="baseconfig. xml">
<import file="${plugins. root}\pclint\macro\pclint. xml"/>
```

(2)根据在 baseconfig. xml 文件中配置的 sourcesPathRefID 参数,对执行 PC-Lint 检查的模块代码路径进行检查。

(3)确定进行检查的代码路径之后,执行 pclint\macro\pclint. xml,调用 pclint\tool\LINT-NT. exe 对模块进行 PC-Lint 检查。

执行脚本代码如下所示:

```
<property name="pclint. executable" location="${pclint. root}/LINT-NT. exe"/>
<apply dir="@{workingdir}" executable="${pclinr. executable}">
<arg value="-b"/>
<optionlist/>
<arg value="i&amp;quot;@{workingdir}&amp;quot;"/>
<arg value="i&amp;quot;@{lnt}&amp;quot;"/>
<arg value="@{lnt}"/>
<pathrefid="@{sourcesPathRefID}">
</apply>
```

### 3.5 检查结果的处理

将所有模块的 PC-Lint 任务都配置完毕之后,启动执行 ICP-CI 页面执行构建工程,工程执行完毕之后,ICP-CI 工具的页面上可以看到所有模块执行 PC-Lint 工具的检查结果。同时 PC-Lint 对所有模块的检查结果通过在 ICP-CI 构建工程上配置的邮件主送人和抄送人,以邮件形式发送。主送人通常是模块的开发工程师和持续集成工程师,抄送人是产品经理与各开发组组长,邮件的发送人是 ICP-CI 工具。

## 4 典型案例

某公司有一个软、硬件结合的中型开发项目,总的代码量为二百万行。Linux 环境下采用 C/C++进行软件开发;采用的软件配置管理工具为 ClearCase,版本为 7.0.1;持续集成工具为 ICP-CI;PC-Lint 工具版本

为 8.0.0w。集成到 ICP-CI 工具后,完成对软件所有模块的 PC-Lint 检查。其中,lop 模块和 ls 模块静态检查的扫描结果如表 4 所示。

表 4 lop 模块和 ls 模块进行 PC-Lint 检查的扫描结果

| Name | File | Error | Warning | Information |
|---|---|---|---|---|
| lop | 118 | 10 | 16 | 143 |
| ls | 20 | 2 | 17 | 31 |

File 表示检查出 Error、Warning 以及 Information 所有错误和告警的文件数,如果被检查的代码中没有任何 Error、Warning 以及 Information 错误和告警,则 File 对应的数值应该是零。

Error 表示模块检查出来的 Error 级别告警数量;Warning 表示模块检查出来的 Warning 级别告警数量;Information 表示模块检查出来的 Information 级别告警数量。

PC-Lint 检查报告中详细列举了告警类型、种类与编号,模块的开发工程师查看检查报告后,对检查出告警的模块源代码进行修改与优化。经过对模块的代码修改以及优化之后,再进行 PC-Lint 检查,错误和告警数目清零,从而改进模块的源代码质量。

工作实践表明,PC-Lint 检查有助于及时发现并解决 C/C++源代码的各种缺陷,也便于产品主管了解工作进度和解决存在的问题,进一步提高软件代码质量。

## 5 结束语

PC-Lint 在代码走读和单元测试之前进行检查,可以提前发现程序隐藏错误,提高代码质量,节省测试时间。使用 PC-Lint 的编码规则检查,可以有效地规范软件人员的编码行为。长期的工作实践表明,PC-Lint 检查在 C/C++程序开发过程中发挥了重要作用。PC-Lint 工具集成到持续集成工具 ICP-CI,可以自动完成 PC-Lint 检查,快速地向软件开发人员反馈检查结果,使软件开发人员能够及时修复源代码的缺陷。软件开发过程中 PC-Lint 检查工作做好了,有助于提高产品质量,降低软件开发成本。

**参考文献:**


[1] Reference manual for PC-Lint/FlexeLint(Software Version 8.00 and Later Document Version 8.00)[R]. [s. l.]:Gimpel Software,2001.

[2] Reference manual for PC-Lint/FlexeLint(Software Version 9.00 and Later Document Version 9.00)[R]. [s. l.]:Gimpel Software,2008.

[3] Gimpel J. Software that checks software:the impact of PC-Lint[J]. IEEE Software,2014,31(1):15-19.

[4] Version 9.00 patches and support files[EB/OL]. 2015. http://www.gimpel.com/html/ptch90.htm.

[5] 黄贤君. 基于 INFINEON ULC2/ULC3 平台的 PC-LINT 新方案的设计与实现[D]. 西安:西安电子科技大学,2012.

[6] 周伟明. 软件测试实践[M]. 北京:电子工业出版社,2008.

[7] 蔡建平. 软件测试实验指导教程[M]. 北京:清华大学出版社,2009.

[8] Meyers S. More effective C++[M]. Reading MA:Addison-Wesley,1996.

[9] Meyers S. Effective C++[M]. 3rd ed. Reading MA:Addison-Wesley,2005.

[10] 白 乔,左 飞. 把脉 VC++[M]. 北京:电子工业出版社,2009.

[11] 杨树青,王 欢. Linux 环境下 C 编程指南[M]. 北京:清华大学出版社,2007.

[12] River W. VxWorks 程序员指南[M]. 北京:清华大学出版社,2003.

[13] 陈智育,温彦军,陈 琪. VxWorks 程序开发实践[M]. 北京:人民邮电出版社,2004.

[14] 罗时飞. 敏捷持续集成(CruiseControl 版):高效研发之道[M]. 北京:电子工业出版社,2008.


(上接第 30 页)


IEEE,2011:409-416.

[12] Muja M,Lowe D G. Fast approximate nearest neighbors with automatic algorithm configuration[C]//International conference on computer vision theory and applications. Lisboa,Portugal:INSTICC Press,2009:331-340.

[13] Wu Y,Lim J,Yang M H. Visual tracking benchmark[EB/OL]. [2015-12-14]. http://www.visual-tracking.net.

[14] Zhang K,Zhang L,Yang M H. Real-time compressive tracking[C]//Proceedings of European conference on computer vision. Florence,Italy:IEEE,2012:864-877.

[15] Possegger H, Mauthner T, Bischof H. In defense of color-based model-free tracking[C]//Proceedings of IEEE conference on computer vision and pattern recognition. Boston,USA:IEEE,2015:2113-2120.

[16] Wu Y,Lim J,Yang M H. Online object tracking:a benchmark[C]//Proceedings of IEEE conference on computer vision and pattern recognition. Los Alamitos, USA:IEEE Computer Society Press,2013:2411-2418.

[17] Everingham M,Gool L,Williams C,et al. The pascal visual object classes (VOC) challenge[J]. International Journal of Computer Vision,2010,88(2):303-338.