黑客派

使用 Csmith 测试 C 编译器

作者：Hanseltu

原文链接：https://hacpai.com/article/1577535040436

来源网站：黑客派

原文链接： 使用 Csmith 测试 C 编译器
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></script>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342" data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></ins>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h3 id="前言">前言</h3>
系统软件如编译器无论在小型还是大型软件系统中都有着相当重要的作用，一个编译器的 bug 有可能对软件带来灾难性的影响。因此，测试编译器变得尤为重要。本文将介绍如何使用 Csmith 测常见 C 编译器如 GCC,LLVM。关于其具体实现原理我将另外写一篇文章详细介绍。
<h3 id="什么是Csmith">什么是 Csmith</h3>
Csmith 是一个可以随机产生有效 C 程序的软件，它可通过随机测试（也称为模糊 测试）帮助发现编译器中的错误。在给定实现 C 标准的多个 C 编译器情况，我们通常可以通运行多种不同版本编译器并比较它们的输出来判定是否产生了编译器的 bug。目前 Csmith 已经发现超过 400 个编译器 bug，包括在 GCC 和 LLVM 上的。Csmith 主页见 Csmith.
常见的测试为差分测试，如下图

大概的测试流程分为以下几步：

 第一步，使用 Csmith 生成大量有效 C 程序
 第二步，将同一 C 程序运行在不同版本的编译器上（或者运行在同一编译器的不同编译选项上）
 第三步，比较编译器的输出，如果编译出的结果不同，则说明肯定有错误，取大部分产生的结果正确输出，少数的为错误输出。

<h3 id="安装及使用">安装及使用</h3>
<blockquote>
 使用环境：Ubuntu 18.04
</blockquote>
1 下载源码 （目前最新为 2.40 版）
<pre><code class="highlight-chroma">git clone https://github.com/csmith-project/csmith.gi

</code></pre>
2 编译
<pre><code class="highlight-chroma">cd csmith
cmake .
make
</code></pre>
3 简单生成测试代码
<pre><code class="highlight-chroma">csmith > test.c
</code></pre>
此时在当前目录下即可看见自动生成的 <code>test.c</code> 文件还有一个平台信息文件。
4 使用脚本重复测试 (脚本文件为 test-csmith.sh)
<pre><code class="highlight-chroma">set -e

</code></pre>

```
for ((i=0; i&lt;10000, i=i+1))
do
  csmith &gt; test.c;
  gcc-4.0 -I ${CSMITH_HOME}/runtime -O -w test.c -o /dev/null;
done
```

执行脚本文件

```
./test-csmith.sh
```

此时，Csmith 将会不断产生新程序不断测试，可以将 gcc-4.0 换成其他版本或者加上其他不同编译优化参数进行测试，最后判断输出是否一致。

相关工作

在编译器测试领域还有其他一些代码生成技术，如何生成有效的程序并且快速找到 bug 还是比火并且是有挑战性的。

以下摘自一篇最新的编译器测试综述 《A Survey of Compiler Testing》，感兴趣的朋友可以看原文。

文章中统计了目前编译器测试研究领域的基本情况，主要有一下几个研究热点：

- 如何构造有效的程序输入，程序往往是高度结构化的；
- 如何解决测试的 Oracle 问题，即如何判断一个程序的输出是正确的；
- 如何优化测试的进程，包括时间和空间上的优化；
- 如何处理大量的 crash，人工处理 crash 非常耗时且难度大；
- 如何从已有的编译器测试中学习经验，指导编译器测试的发展。

下图是编译器在以上五个方向中研究论文的数量及总体研究情况。

可以看出，近几年学术界对编译器的测试工作还是越来越感兴趣的。

思考

- 系统软件的缺失是我国的“卡脖子”难题，我国目前也在不断加大对系统软件的投入。包华为的方舟及鸿蒙的研究。
- Csmith 这个工具的缺点可能是这些自动生成的程序和人写的代码还是有一定的差距，比如些特定的函数，如求阶乘，不能有效生成，这类函数的测试如何解决？
-

参考：

1 Csmith 原文 （2011 PLDI <a href="https://link.hacpai.com/forward?goto=http%3A%2F2Fwww.cs.utah.edu%2F%7Eregehr%2Fpapers%2Fpldi11-preprint.pdf" target="_blank" rel="no

ollow ugc">Finding and Understanding Bugs in C Compilers）.
 2 2019 最新编译器试综述 《A Survey of Compiler Testing》.</p>