Gcov记录文档

编辑历史

更新日期	作者	更新内容
2024.02.20	胡益华	Initialization

目录

1. 引言

1.1 概述

即使聪明的编译器可能会优化一小部分不可能执行到的代码片段，实际进程运行时，也不一定会执行到每一行代码。这一部分未被执行到的代码可能有其存在的价值，也有可能永远无法被执行。

有时适当的统计代码覆盖率是有必要的，它可以帮助工程师们直观地了解源代码中每行代码的执行与否。尽管代码覆盖率高不能说明代码的质量就是高的，但是极低的代码覆盖率说明代码可能确实存在一些问题。

假设有一段代码，它的每一行都有存在的价值，在不同情况下都会被执行到，那么此时统计代码覆盖率还有意义吗？答案是肯定的。比如单元测试时，工程师肯定希望自己的测试用例尽可能多地覆盖到源代码，甚至精确到某一行，此时要怎么做？如果是我，那么第一反应可能是加打印吧。当然聪明的工程师们不会使用加打印这种方式，为此，他们开发出了一系列覆盖率测试工具。对于 C/C++ 而言，当前主流的覆盖率检测工具有 Gcov、BullseyeCoverage 等。本文档主要对 gcov、gcovr 的使用做一些记录。

1.2 gcov 和 gcovr

gcov 是 gcc 自带的工具，用于代码覆盖率的检测，一般来说，gcov 和 gcc 一起工作，帮助工程师们分析代码覆盖率，它并不适用于其它编译器。

尽管 gcov 提供了分析代码覆盖率的功能，不过它生成的是一些纯文本文件，所以结果可读性上可能会差一些，也没有那么直观。为此，Lcov、Gcovr 等工具应运而生，它们可以将覆盖率报告从文本形式转化成 html、xml 等格式，使得覆盖率报告变得清晰易读。

gcovr 是一款开源的软件。gitbub 地址如下：

https://github.com/gcovr/gcovr

官方使用手册地址如下：

https://gcovr.com/en/stable/

gcovr 不仅适用于 gcc，对 clang 也同样支持。同时支持 Linux、Windows 等系统。如果环境中有 python，gcovr 的安装会格外的简单，执行如下命令即可：

```
pip install gcovr
```

2. 覆盖率统计过程

2.1 插桩

在编译链接参数中添加 gcov 选项后，编译器会做一些额外的操作。基本原理是通过在代码中插入计数器来统计覆盖率，也就是所谓的插桩。

一般来说，插桩不是直接在源代码中插入，而是采用在编译的中间文件插桩的方式，即在汇编文件中插入计数器以及收集这些计数器信息的模块。进程结束后，收集到的信息会被写入结果文件中。

2.2 覆盖率文件

在编译链接参数中添加 gcov 选项后，编译完成后会生成一些后缀为.gcno 的文件，这些文件记录的是源文件和插入的计数器位置的映射关系，一般.gcno 文件和.o 目标文件放在同一目录。

在进程运行后，会生成一些后缀为.gcda 的文件，这些文件记录了代码执行的情况，包括基本模块被执行的次数等信息。

将.gcov 和.gcda 文件结合分析，就能生成相关的覆盖率报告。

2.3 编译链接参数

2.3.1 编译

编译时，为了生成.gcno 文件，需要添加以下参数：

```
CFLAGS    += -g -O0 --coverage
```

若仅在 LDFLAGS 添加覆盖率链接参数，CFLAGS 没有添加覆盖率编译参数，那么链接时可能不会报错，但是将没有.gcno 文件生成，可执行文件运行后也不会有.gcda 文件生成。

2.3.2 链接

链接时，同样需要添加--coverage 参数：

```
LDFLAGS   += --coverage
```

若仅在 CFLAGS 添加覆盖率编译参数，LDFLAGS 没有添加覆盖率链接参数，那么链接的时候会报错。

2.3.3 参数说明

--coverage 参数可以看作-fprofile-arcs 和-ftest-coverage 两个参数的结合。无论是编译还是链接，上述两种参数都可以互相替换，不必纠结。

实际上，编译参数 CFLAGS 添加-fprofile-arcs 和-ftest-coverage；链接参数 LDFLAGS 添加-fprofile-arcs 即可。但使用时建议直接使用--coverage 参数，清晰明了，避免错误。

2.4 结果获取

开启覆盖率测试后，运行可执行文件，会生成.gcda 文件。一般这些.gcda 文件会自动生成到和它对应的.gcno 目录下，接着用 gcovr 便可以分析覆盖率结果。

但当编译环境和文件执行环境不同时，.gcda 文件生成的默认路径还是和.gcno 一样，但是这样是行不通的。举个例子，我在计算机 A 上编译生成了可执行文件 target，.gcno 文件的路径是/home/usadayu/gcov_demo/obj/*.gcno。接着我将 target 文件放到了计算机 B 上执行，此时.gcda 文件默认也会尝试着生成到/home/usadayu/gcov_demo/obj/目录下，然而计算机 B 上对某些目录是否有写权限就很难说了，或者计算机 B 上根本没有/home/usadayu/目录。

所以如果程序是在不同环境运行的，运行前需要配置 gcov 的环境变量：GCOV_PREFIX 和 GCOV_PREFIX_STRIP。

(1) GCOV_PREFIX 表示文件执行时的环境中，生成.gcda 文件的前缀；

(2) GCOV_PREFIX_STRIP 表示在原先默认生成的.gcda 文件路径中，去掉的目录层数。

上述文字描述可能比较抽象，还是以计算机 A 和 B 举例说明：

假设 gcno 文件的路径是计算机 A：/home/usadayu/gcov_demo/obj/*.gcno；

可执行文件执行的路径是计算机 B：/share/usadayu/target。

现在我希望.gcda 文件生成到计算机 B：/share/usadayu/gcov_result/*.gcda，那么 gcov 环境变量配置如下：

```
export GCOV_PREFIX=/share/usadayu/gcov_result/
export GCOV_PREFIX_STRIP=4
```

GCOV_PREFIX_STRIP=4 表示去掉/home/usadayu/gcov_demo/obj/这 4 层目录。

又比如我的配置如下：

```
export GCOV_PREFIX=/share/usadayu/gcov_result/
export GCOV_PREFIX_STRIP=1
```

表示只去除掉了 1 层目录，也就是/home/这层目录，此时 gcda 文件路径为：/share/usadayu/gcov_result/usadayu/gcov_demo/obj/*.gcda。

上述两个环境变量结合使用，生成.gcda 文件后可以将它们拷贝到原编译环境，再用 gcovr 等工具进行分析。

3. gcovr 使用

3.1 gcovr 命令

一般只要.gcno 和.gcda 文件生成正常，那么使用如下命令就可以打印出覆盖率的一些信息：

gcovr -r .

运行 gcovr 时，配置参数选项可以实现一系列不同的报告呈现。

(1) --gcov-executable

选择 gcov 的可执行文件，不选择一般默认使用本地 x86 的 gcov。注意这里是可执行文件 gcov 而不是 gcovr，使用 which 命令可以看到 gcov 在 gcc 目录下。

(2) --gcov-ignore-parse-errors

告诉 gcov 忽略在分析过程中发现的错误，继续执行并生成覆盖率报告，即使存在某些解析错误。

(3) --exclude-unreachable-branches

排除不可抵达的分支。实际使用时我暂时没有尝试出此参数的具体效果，使用时尽量加上即可。

(4) --print-summary

在终端打印一个大致的统计数据。

(5) --html

生成 html 格式的覆盖率报告。

(6) --html-details

为每个覆盖率测试的文件生成详细的 html 格式报告。

(7) --filter

选在需要检测覆盖率的代码路径，添加后只检测--filter 指定的目录或文件。

(8) --exclude

排除需要检测的目录或文件。

3.2 gcovr 注意事项

3.2.1 html

(1) --html 参数生成的报告仅包含每个文件的行覆盖率、函数覆盖率、分支覆盖率等。

(2) 如果需要生成每个文件中的具体执行信息，那么--html-details 参数会是

更好的选择，当然，--html-details 参数也会生成一份综合性的报告，和--html 生成的报告一样，也就是说--html 生成内容是--html-details 生成内容的真子集。

3.2.2 --filter and --exclude

--filter 和--exclude 参数后的文件或目录都可以通配符或正则表达式等。因为.符号表示当前目录，所以如果要匹配名字中含.的文件，需要在.前面添加\转义。如*.h 文件，那么就是*\.h。举几个简单的例子：

(1) 排除绝对路径下对所有头文件的检测，写成：

```
--exclude="/.*\.h"
```

(2) 排除相对路径下对所有头文件的检测，写成：

```
--exclude=".*\.h"
```

(3) 排除所有头文件和桩库的检测，可以用|分隔，写成：

```
--exclude=".*\.h|/home/$USER/gcov_demo/src/stub/"
```