计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(2):68−74 [doi: 10.15888/j.cnki.csa.006776]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Linux 平台下代码覆盖率报告自动化输出设计①

石佳琦[1], 陈　鹏[2]

[1](武汉邮电科学研究院, 武汉 430010)
[2](烽火通信科技股份有限公司, 武汉 430010)
通讯作者: 石佳琦, E-mail: 472377487@qq.com

摘　要: 白盒测试中经常用到覆盖率测试. 轻量级覆盖率测试工具 GCOV 在使用上具有操作复杂繁琐的弊端. 本文所述的设计基于 GCOV 覆盖率测试的原理, 依据 shell 脚本批处理的特点将各部分操作封装为脚本工具; 通过 Expect 脚本工具对上一操作是否成功进行断言, 实现各脚本工具调用的联动. 通过该设计进行覆盖率报告输出比传统的操作节省了大量时间. 同时, 本设计具有操作简易、便于移植的特点. 极大的减少了工作人员重复的操作, 提高了软件开发测试的效率.

关键词: Linux; 分布式系统; 覆盖率; 脚本; 自动化

引用格式: 石佳琦,陈鹏.Linux 平台下代码覆盖率报告自动化输出设计.计算机系统应用,2019,28(2):68–74. http://www.c-s-a.org.cn/1003-3254/6776.html

Design of Automatic Output of Code Coverage Report under Linux Platform

SHI Jia-Qi[1], CHEN Peng[2]

[1](Wuhan Research Institute of Posts and Telecommunications, Wuhan 430010, China)
[2](Fiberhome Telecommunication Technologies Co. Ltd., Wuhan 430010, China)

Abstract: Coverage testing is often used in white box testing. The lightweight coverage testing tool GCOV has the disadvantage of operating complicatedly. The design described in this paper is based on the principle of GCOV coverage testing. According to the characteristics of batch processing of shell script, all parts of the operation are encapsulated as script tools. Through the Expect script tool asserting whether the last operation is successful or not, to achieve all script tools being excuted automatically. The output of coverage report by this design saves a lot of time than traditional operation. At the same time, this design has advantages of operating simplely and transplanting easily. It greatly reduces the repeated operation of programmers and improves the efficiency of software development and testing.

Key words: Linux; distributed system; coverage; script; automation

覆盖率对于软件测试有着非常重要的作用. 通过代码覆盖率, 可以发现程序的未测试部分, 估计程序中哪段代码最耗时, 帮助开发人员创建更高效、更快的运行代码, 提高代码质量.

GCOV 是一个 GNU 的本地覆盖测试工具[1], 配合 GCC 共同实现对 C 或者 C++文件的语句覆盖和分支覆盖测试. 而在 Linux 环境下, shell 脚本的魅力大放异彩, 通过带参数的指令集, 可以实现许多的功能. Expect 脚本工具可以实现交互式任务, 而无需人的干预. Cygwin 工具, 更是为我们提供了一种在 Windows 环境下类 Linux 的环境. 本文基于 GCOV 工具覆盖率生成流程, 配合上述软件工具完成 Linux 平台下代码覆盖率报告自动化输出设计[2,3].

① 收稿时间: 2018-08-06; 修改时间: 2018-09-05; 采用时间: 2018-09-20; csa 在线出版时间: 2019-01-28

1 GCOV 覆盖率原理与过程

1.1 GCOV 工作原理

基本块 BB (Basic Block) 是程序中一个顺序执行的语句序列, BB 中的所有语句的执行次数一定相同, 一般由多个顺序执行语句后跟一个跳转语句组成. ARC 分支, 从一个 BB 到另一个 BB 的跳转记为一个 ARC.

如图 1 所示. 如果跳转语句是有条件的, 就产生了一个分支 (ARC), 该基本块就有两个基本块作为目的地. 如果把每个基本块当作一个节点, 那么一个函数中的所有基本块就构成了一个有向图, 称之为基本块图. 只要知道 BB 或 ARC 的执行次数就可以推算出所有的 BB 和所有的 ARC 的执行次数. GCOV 根据 BB 和 ARC 的统计情况来统计各 BB 内各行代码执行情况, 从而计算整个程序的覆盖率情况[4].

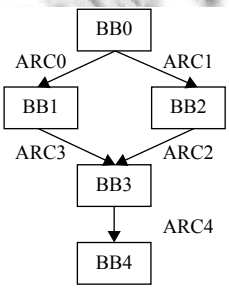

图 1 基本块图

1.2 GCOV 收集覆盖率信息流程

主要工作流程见图 2.

1) 编译前, 在编译器中加入编译器参数-fprofile-arcs-ftest-coverage.

2) 源码经过编译预处理, 在生成汇编文件的阶段完成插桩. 生成可执行文件, 并且生成关联 BB 和跳转次数 ARC 的*.gcno 文件.

3) 执行可执行文件, 在运行过程中“桩点”负责收集程序的执行信息[5].

4) 生成具有 BB 和 ARC 的执行统计次数等数据信的*.gcda 文件.

5) 通过 lcov 和 genhtml 可将*.gcno、*.gcda 中的统计信息图形化, 生成具体的报告文档.

2 设计方案及模型

由于实际运行环境为嵌入式分布式设备, 且编译环境和测试环境分别在不同的服务器上, 我们采用模块化设计, 将各功能的实现模块化, 便于后续的管理优化, 降低各阶段执行的耦合率, 提高覆盖率自动化测试的时效性.

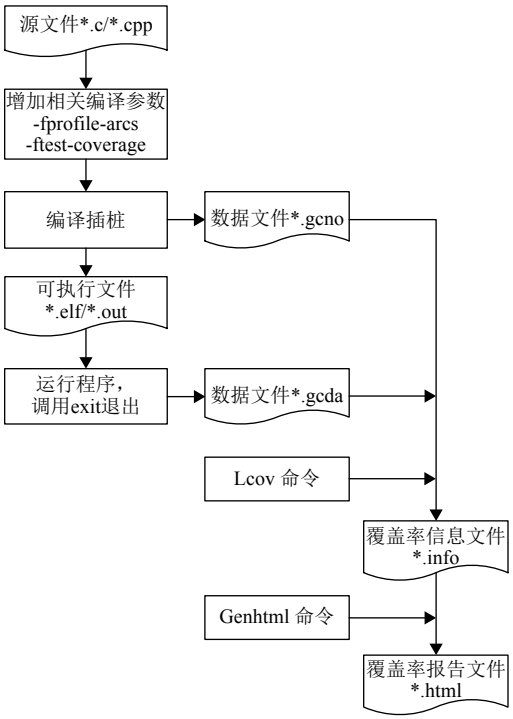

图 2 GCOV 收集覆盖率信息流程图

2.1 设计方案介绍

整体环境分为四部分: 编译环境、运行环境、服务器端、用户端.

编译环境: 主要实现不同模块软件源代码编译打包, 完成覆盖率报告的生成.

运行环境: 软件运行的实际环境. 通过更新软件实际运行, 实现代码在具体目标终端设备机上 运行的代码覆盖率实际情况.

服务器端: 工程测试人员操作平台, 主要完成源代码编译的操作的控制、更新代码在目标终端设备机上运行情况的控制以及代码自动化覆盖率测试及输出的.

用户端: 工程测试人员个人 PC 覆盖率报告文档的目标用户, 可以设置多个.

2.2 系统结构拓扑图

图 3 为系统结构拓扑图.

2.3 软件工具介绍

Cygwin 是一个在 Windows 平台上运行的类 Linux 模拟环境, 它提供一个 UNIX 模拟 DLL 以及在其

上上层构建的多种可以在 Linux 系统中找到的软件包.

Expect 是一种免费的编程工具语言, 用来实现自动交互式任务进行通信, 而无需人工值守. 它是一个用来实现自动交互功能的软件套件.

在当前 Windows 工作服务器安装 Cygwin 软件, 在安装时勾选除 Linux 环境一些常见的软件包外, 还需勾选基于 TCL 的 Expect 软件包, 配置支持远程远程登录的 ssh 等服务[2].

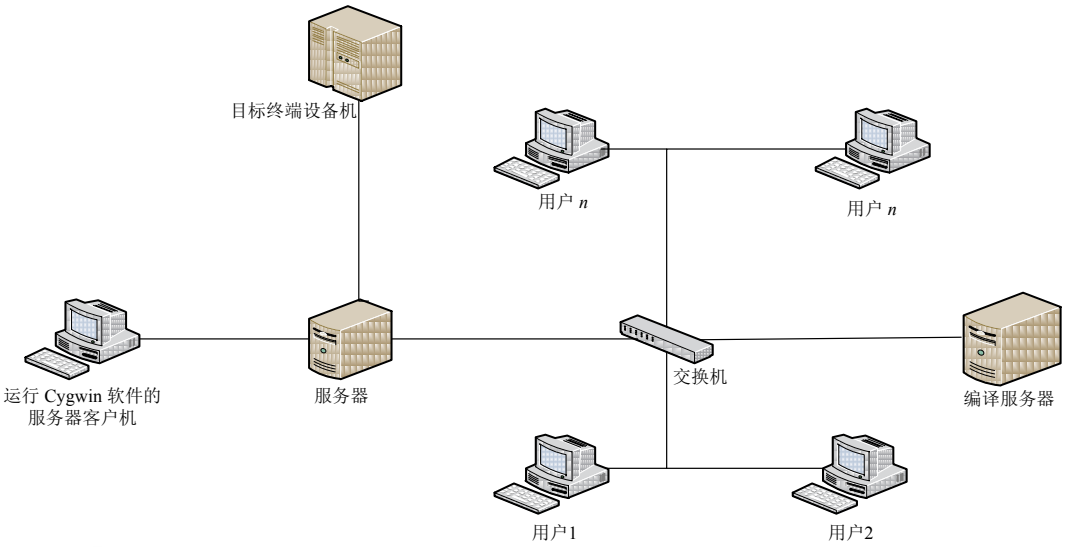

图 3 系统结构拓扑图

2.4 各模块设计目的与实现的功能

2.4.1 服务器端设计目的与实现的功能

由于程序需要在终端设备上运行, 且测试环境和编译环境在不同的服务器上, 这就需要我们设置一个中心节点可以有效的将各模块和环境联系起来, 以实现设计的完整性和操作连贯性.

不同用户可以远程到服务器端, 依据 GCOV 覆盖率报告生成的步骤, 通过调用封装好脚本工具实现如下功能:

自动登录编译环境完成编译, 拷贝可执行文至服务器端的用户目录;

拷贝可执行文件至测试环境, 并将生成的 *.gcda 文件拷贝至用户目录;

自动检查用户目录存在*.gcda 文件时, 拷贝 *.gcda 文件至编译环境对应目录, 执行 lcov 和 genhtml 完成可视化覆盖率报告的生成

自动检查在编译环境对应目录下是否存在生成的覆盖率报告, 并拷贝至服务端用户目录.

2.4.2 编译环境设计目的与实现的功能

大型的 C 工程的编译都有各自的 makefile 和编译脚本. 编译时工程开发人员通过键入不同的参数来编译外链库文件或者可执行文件.

该模块设计主要目的是生成覆盖率数据文件 *.gcno. 通过修改编译脚本实现插桩编译以及外链库和可执行文件的单一编译或混合编译. 通过判断是否带 GCOV 参数来控制是否进行插桩编译, 即使在不进行覆盖率编译的情况下, 也可以实现普通编译及文件拷贝, 减少了开发人员与编译服务器的交互. 该模块主要响应来自“服务器端”开发人员键入命令, 对编译环境下常用编译文件、脚本的解析执行, 完成软件的编译拷贝工作. 将参数提取, 根据工程开发人员在编译模块脚本键入不同的参数组合实现单一库文件或可执行文件的编译, 以及库文件可执行文件的混合编译. 服务器端不编译服务器交互过程见图 4.

2.4.3 运行环境设计目的与实现的功能

运行环境设计的主要目的是生成覆盖率数据文件 *.gcda. 运行可执行文件, 在捕捉到外部信号时, 对程序进行 exit. 检查是否存在*.gcda 文件并通过服务器端转存到编译环境.

2.4.4 用户侧设计目的与实现的功能

用户侧为远程到服务器端的客户端. 是开发人员工作的物理环境.

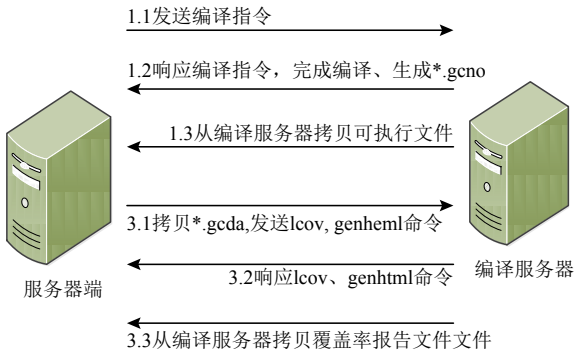

图 4 服务器端不编译服务器交互过程

3 脚本工具封装及流程说明

3.1 各脚本工具封装

3.1.1 流程提示工具

main.sh 参数: (0~9 的数字 或无)

功能说明:

主要通过 echo 命令对整个流程进行索引, 指导流程. 参数: 一个数字. 不带参数时输出整个流程提示, 已数字为流程序号索引; 带参数时, 输出该数字对应流程调用的脚本及示例.

脚本示例:

main.sh 1

3.1.2 拷贝工具

copy.sh 参数: 1) 拷贝环境; 2) 需要拷贝的文件; 3) 目标环境; 4) 目标路径.

功能说明:

完成对本地文件拷贝到目标环境, 或从目标环境拷贝文件到本地. 其中目标环境可以是运行环境或用户端.

功能实现:

远程拷贝时需要输入用户名和密码, 以及路径. 由于我们拷贝文件的目录比较固定, 可以将这些静态数据写成配置文件, 而动态的拷贝环境, 文件名等作为脚本工具的参数.

local_to_dst 主要完成了本地到目标环境的远程拷贝. 将 scp 命令“嵌入”到 Expect 脚本中, 对登录用户写静态配置, 当目标环境提示输入密码时, 将提示语“password”作为 Expect 响应关键字, 此时将配置好的静态密码通过 Expect 中 send 命令发送, 则实现了文件自动拷贝.

同理 dst_to_local 实现了反向的功能.

脚本示例:

copy.sh local shell_shell_gcov dst board1

3.1.3 校验工具

check.sh 参数: 1) 校验环境; 2) 工作目录; 3) 文件类型; 4) 文件名称; 5)user_name; 6)date.

功能说明:

完成对目标环境下目标文档是否存在进行校验, 完成不同拷贝动作.

功能实现:

调用本地或拷贝到运行环境和编译环境 shell_for_gcov 下的 find_the_file.sh , 将目标文件校验结果写入校验文件, 并拷贝到服务器端建立的用户目录下. 我们对不同的目标文件都设有一个以文件类型结尾的校验结果文件, 通过这个文件写入的内容判断是否存在目标文件, 是否可以进行下一步的拷贝动作.

脚本示例:

check.sh root gcov gcda gcda-0506 sjq 20180506

3.1.4 编译工具

complex_compile.sh

参数: 1) 编译类别; 2) 编译目录; 3) 工作路径; 4) 版本信息; 5) user_name; 6) date.

功能说明:

通过编译类别远程不同的编译服务器, 通过编译目录执行不同的编译脚本, 加版本信息对每次编译进行区分.

功能实现:

首先对编译 makefile 修改增加和覆盖率相关的编译和链接选项.

其次在当前前 Windows 服务器选定一个工作路径, 以以此路径为基路径, 调用 mkdir.sh 脚本, 两个参数 1)、人名, 2) 日期 , 以 user_name 为基路径下一级目录, 日期作为一级目录下二级目录, 用于存放相关文件. 检查基路径下所有一级目录名, 若不存在则新建, 若存在则进入该目录, 检查一级目录下二级目录名, 如不存在则新建, 若存在则输出提示, 新建的目录名后加 bak 后缀以作区分.

通过 Expect 交互脚本完成对可执行文件以及可执行文件依赖的外链库进行自动化编译. 软件包内的不同可执行文件是否需要编译可以通过修改配置文件完成, 编译时将编译配置文件内的参数传入到编译脚本. 如果外链的库需要编译, 则先编译外链库再将库文件

拷贝到编译可执行文件的 lib 库目录下编译[6,7].

编译完成后拷贝到服务器端的建立用户目录下, 完成软件包的解压, 将可执行文件拷贝出.

脚本示例:

complex_compile.sh ccu " " 686214_V2R5_new 128 sjq 20171207

3.1.5 备份工具

backup.sh 参数: 1) 备份环境; 2) 备份文件类型; 3) 是否恢复操作标志; 4) user_name; 5)date.

功能说明:

完成不同环境上容易变动的文件的备份, 以便环境恢复出错时可以降回到以前版本.

功能实现:

主要在备份环境上不同类型的文件备份目录下以 user_name 为基路径下一级目录, 日期作为一级目录下二级目录. 将文件拷贝只该目录下. 需要恢复环境时, 将该目录下的文件拷回原目录下.

脚本示例:

Backup.sh dst exe 0 sjq 20180506

功能说明:

完成对本地文件拷贝到目标环境, 或从目标环境拷贝文件到本地. 其中目标环境可以是运行环境或用户端.

3.2 主要功能实现流程

图 5 为本系统工作的主要流程.

流程简述:

1) 调用 mkdir.sh 输入 user_name 和 date 信息, 在服务器端建立用户目录用于存放文件;

2) 调用 compile.sh 完成外链库或软件的编译, 编译成功通过控制台会输出关键字“success”;

3) 通过 expect 交互脚本语言做出匹配该关键字的操作, 即调用服务器上 shell_for_gcov 下的 find_the_file.sh 查找是否存在编译好的文件并写校验文件, 然后将校验文件拷贝到服务器端. 此时, expect 脚本会自动调用 check.sh 读取用户目录下的校验文件, 如果读取可以拷贝, 后将带有覆盖率编译的软件拷贝至用户目录下, 解压软件将可执行文件取出;

4) 将运行环境上的可执行文件备份, 将用户目录的新文件拷贝运行环境运行;

5) 登录运行环境系统设置覆盖率相关的环境变量, 运行程序;

6) 触发指令操作, 覆盖代码执行, 待程序退出此时会在环境变量设置的目录下生成存放同名 C 文件的 GCDA 文件目录;

7) 校验运行 GCDA 文件是否生成, 并拷贝至编译服务器上的编译目录;

8) 在编译环境执行覆盖率命令, 此时会将覆盖率的信息以 html 格式的文件输出;

9) 校验是否生成覆盖率报告, 并通过 mail 命令拷贝到用户目录.

4 测试验证及优化建议

4.1 测试结果

本设计通过作者本人实际工作中进行覆盖率测试的经验和经历提炼而来. 在进行覆盖率统计时, 很多准备工作都是是单调重复的操作[8]. 由最开始 shell 脚本设计的批处理拷贝, 在实践中不断完善, 参考覆盖率生成的步骤, 将其设计成脚本工具, 在实际中的确减少了人机的频繁交互, 节省了时间, 提高了工作效率.

4.2 优化建议

1) 该设计并不是完全自动的, 部分修改还需人工操作, 如:

需要在测试 main 函数中增加信号捕捉函数, 以便后台程序可以 exit;

需要在运行环境中手动设置环境变量, 该步骤可以通过脚本实现;

程序的运行需要手动操作, 该步骤也可以通过脚本实现;

2) 本设计前提是在运行环境和编译环境可以登录的情况下进行的, 没有对环境的可用性进行检查, 可以通 ping 命令配合过脚本实现对环境的可用性检查;

3) 日志记录并没有保存文件, 可以通过重定向的方式将控制台打印按一定格式存文件;

4) 本设计的分支测试需要外部命令触发, 可以通 gtest 工具对增加测试用例, 配合 GCOV 实现对代码的全覆盖, 给出更全面的报告;

5) 目前的覆盖率是全量的覆盖率, 可以优化实现增量代码的覆盖率.

还有很多不足之处和待优化的地方需要改进.

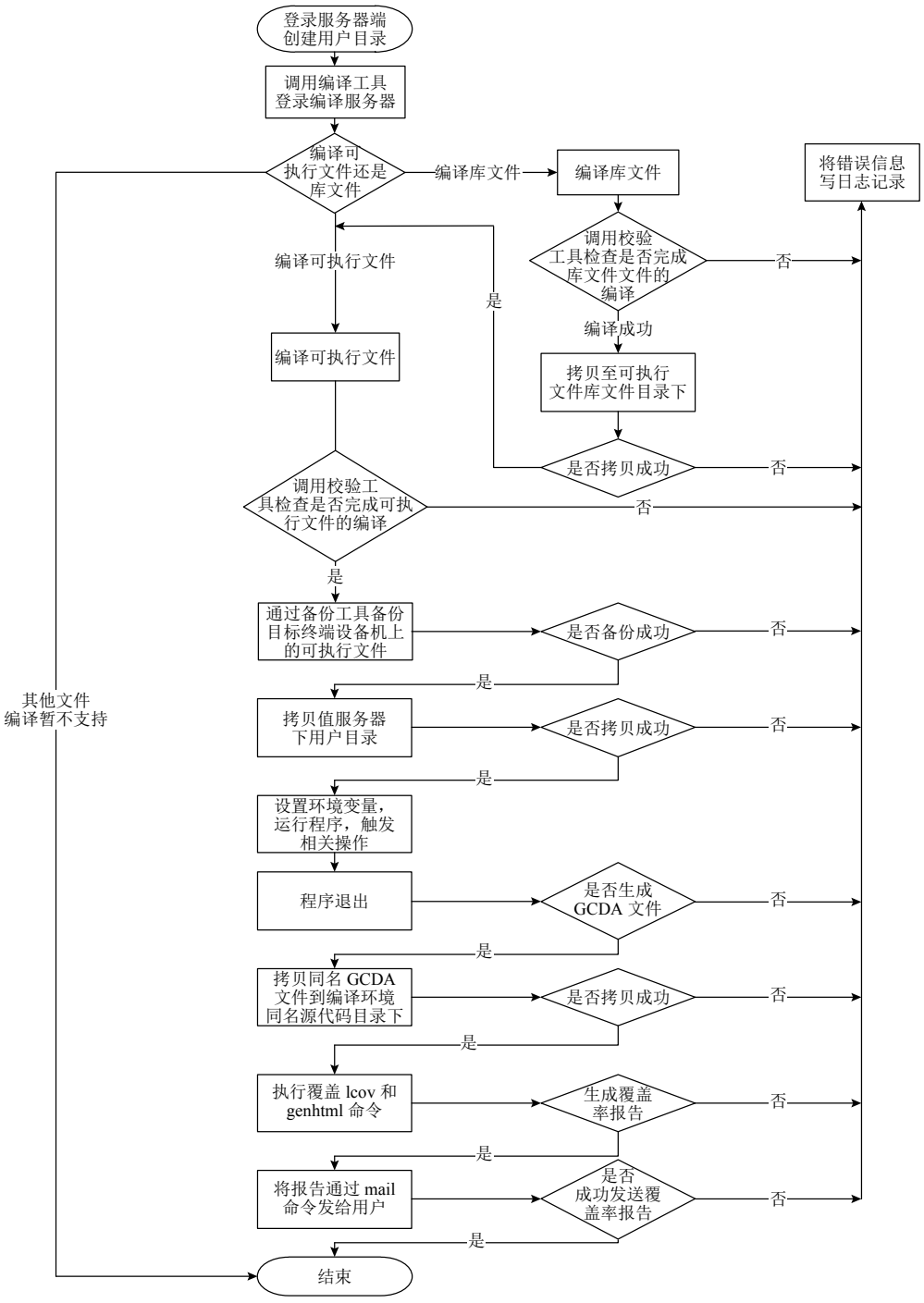

图 5 代码覆盖率报告自动输出流程图

4.3 测试验证

1) 如图 6, 首先进入服务器端工作路径, 调用提示工具提示操作步骤;

2) 调用目录创建工具, 完成用户目录的创建 (见图 7);

3) 调用编译工具完成, 软件的编译 (见图 8);

4) 在运行环境完成*.gcda 文件生成, 检验后拷贝到服务器端, 此时调用覆盖率自动输出工具, 会完成覆盖率报告的自动输出 (见图 9), 并拷贝的服务器端用户目录下, 见图 10.

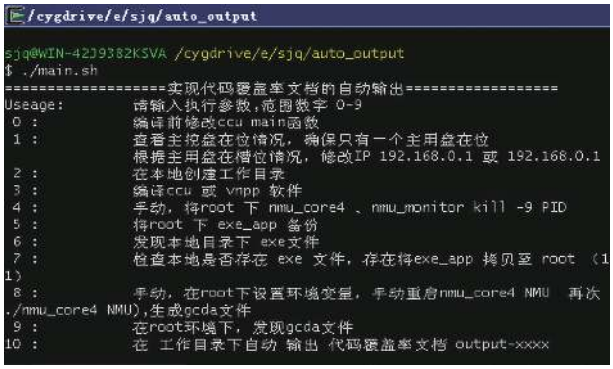

图 6 提示工具使用样图

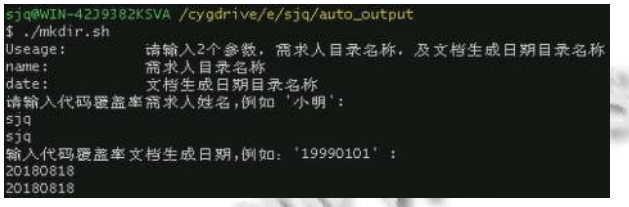

图 7 目录创建工具使用样图

图 8 编译工具使用样图

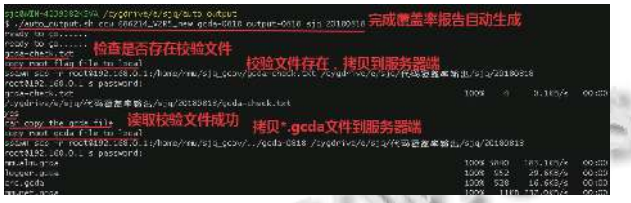

图 9 覆盖率报告自动输出工具使用样图

图 10 拷贝到用户目录下的覆盖率报告及各种校验结果文件

如图 11 所示, 我们可以看到整个源码的覆盖率情况, 和函数覆盖百分比. 点击某个 C 文件我们可以看到每一行代码执行的频率, 还有各函数的覆盖情况以及内部分支执行的情况.

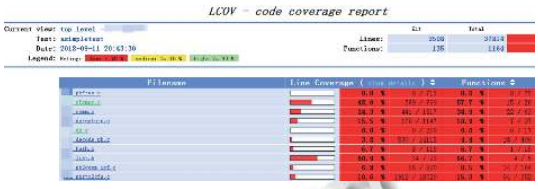

图 11 生成的覆盖率报告

5 结束语

本文介绍了一种 Linux 平台下代码覆盖率自动化输出设计. 首先对本设计依据原理进行了介绍, 然后对系统整体的架构进行了介绍, 以及各模块的设计目的及实现功能进行了描述, 最后依据该设计进行测试, 并结合实际操作所遇到的问题, 指出了本设计的不足之处和优化建议.

本设计是作者在进行 Linux 下覆盖率测试时, 发现基于 GCOV 的覆盖率测试具有操作繁琐单调、耗时的特点, 为解决这一问题对覆盖率输出做出了改进优化. 与原有 GCOV 覆盖率输出相比, 本设计具有搭建方便、操作简单, 节省时间的的优点. 同时通过修改编译脚本和相关路径及 IP 的配置文件, 可实现移植. 在现有的条件下极大的提供高了覆盖率报告的输出效率, 为工程开发人员提供了有效依据, 节约了大量时间.

参考文献

1 李超, 史晓华, 王斐. 一种轻量级的代码分支覆盖率检测方法: 中国, CN106294163A. 2017-01-04.
2 张世伟. 数据通信设备自动化测试框架设计与实现. [硕士学位论文]. 成都: 电子科技大学, 2017.
3 姜文, 刘立康. 基于持续集成的 C/C++软件覆盖率测试. 计算机技术与发展, 2018, 28(3): 37–41, 46. [doi: 10.3969/j.issn.1673-629X.2018.03.008]
4 周雷. 嵌入式代码覆盖率统计方法. 计算机应用与软件, 2014, 31(5): 326–327. [doi: 10.3969/j.issn.1000-386x.2014.05.083]
5 毛养红. 自动化单元测试的测试用例扩展对桩代码的优化. 当代教育实践与教学研究, 2016, (5): 212–213, 211. [doi: 10.3969/j.issn.2095-6711.2016.05.186]
6 蒋云, 赵佳宝. 自动化测试脚本自动生成技术的研究. 计算机技术与发展, 2007, 17(7): 4–7. [doi: 10.3969/j.issn.1673-629X.2007.07.002]
7 凌永发, 张云生, 郭秀萍. 软件测试自动化中的脚本技术. 云南民族学院学报 (自然科学版), 2002, 11(1): 544–548. [doi: 10.3969/j.issn.1672-8513.2002.01.006]
8 李斌, 陈榕. 和欣编程环境中进行单元测试覆盖率分析的方法. 福建电脑, 2008, 24(6): 1–2, 4. [doi: 10.3969/j.issn.1673-2782.2008.06.001]