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.

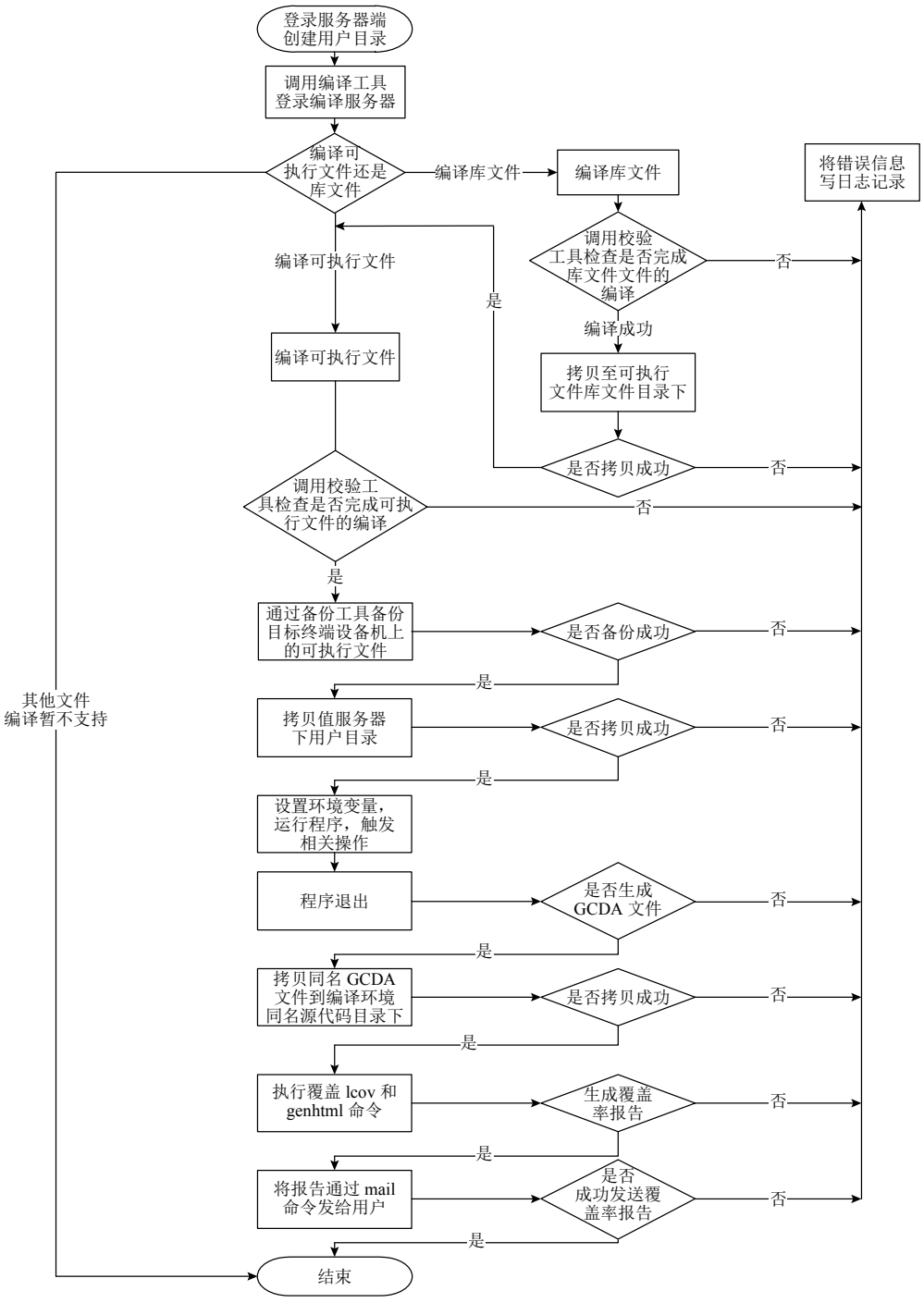

图 5 代码覆盖率报告自动输出流程图

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.

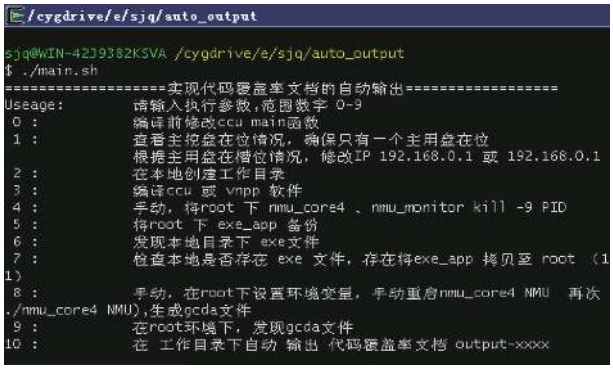

图 6 提示工具使用样图

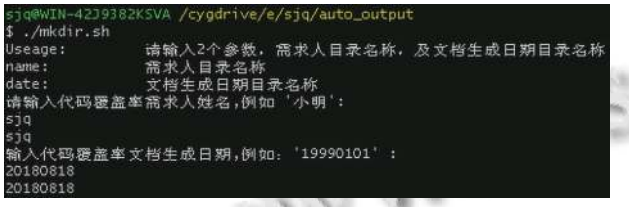

图 7 目录创建工具使用样图

图 8 编译工具使用样图

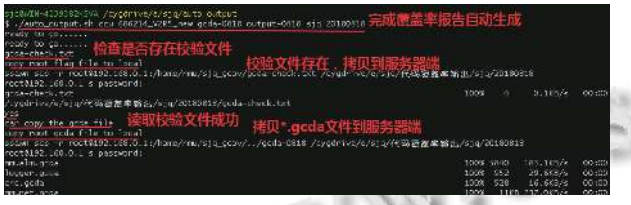

图 9 覆盖率报告自动输出工具使用样图

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.

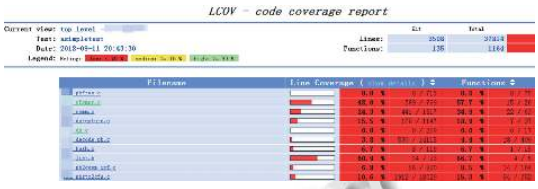

图 11 生成的覆盖率报告

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]