# GCOV代码覆盖方法

韦体东

## 编写Makefile(编译环境)

Makefile里面，加入编译项：

-g -fprofile-arcs -ftest-coverage

链接项：

-lgcov

考虑到后续代码覆盖可能需要调试，建议加入-O0和-g选项，但是不建议定义DEBUG，因为assert会导致程序abort。

我们的编译环境下，可以直接加入如下模板：

RTSERVER = youraddr

RTPASSWD = yourpassword

RTDIR = ${PWD}

RTRUNCMD = /sf/etc/init.d/$(TARGET)

rcovs:

make clean

sshpass -p ${RTPASSWD} ssh root@${RTSERVER} "${RTRUNCMD} stop"

make GCOV\_FLAGS="-g -fprofile-arcs -ftest-coverage" GCOV\_LIBS="-lgcov"

sshpass -p ${RTPASSWD} ssh root@${RTSERVER} "rm -rf ${RTDIR}/\*; mkdir -p ${RTDIR}; mkdir -p ${RTDIR}/res"

sshpass -p ${RTPASSWD} scp ${RTDIR}/\*.gcno ${RTDIR}/\*.c root@${RTSERVER}:${RTDIR}

sshpass -p ${RTPASSWD} scp ${RTDIR}/$(TARGET) root@${RTSERVER}:/sf/bin

sshpass -p ${RTPASSWD} ssh root@${RTSERVER} "${RTRUNCMD} start"

rcove:

sshpass -p ${RTPASSWD} ssh root@${RTSERVER} "${RTRUNCMD} stop; lcov --directory ${RTDIR} --capture --output-file ${RTDIR}/app.info; genhtml -o ${RTDIR}/res ${RTDIR}/app.info; cd ${RTDIR}; tar -zcf report.tar.gz res"

@echo retrieve from ${RTDIR}/report.tar.gz

rcovc:

sshpass -p ${RTPASSWD} ssh root@${RTSERVER} "${RTRUNCMD} stop; rm -rf ${RTDIR}"

其中rcovs是remote coverage start的缩写，rcove是remote coverage end的缩写，rcove是remote coverage clean的缩写。

## 初始化覆盖环境(编译环境)

RTSERVER和RTPASSWD变量，为运行时的服务器IP和SSH密码，需要在Makefile里面指定，然后执行 make rcovs即可自动编译并部署。

## 运行测试案例覆盖(运行环境)

在运行时，运行APP，如果需要，可以使用调试器，主动调用逻辑或者修改变量

# gdb --args /sf/bin/docker-status -v

(gdb) b channelInit

Breakpoint 1 at 0x40be32: file channel.c, line 158.

(gdb) r

Starting program: /sf/bin/docker-status -v

[Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

15:56:54.912 43634 D cluster.c:374 | add new cluster {id:'all' version:1 update:'01-01 08:00:00' master:0/0 members:0 []} ok

Breakpoint 1, channelInit (clusters=0x6429b0, evbase=0x6240a0) at channel.c:158

158 struct Channel\* channel = calloc(1, sizeof(struct Channel) + CH\_PKG\_DATA\_LEN\_MAX);

(gdb) n

159 if (NULL == channel) {

(gdb) set channel=0

(gdb) call ds\_strcpy\_unbroken(gContext->channel->recvBuffer, 44, "444sss")

$6 = 6

(gdb) call ds\_strtol("44", &gContext->channel->recvBufferLen)

$7 = 0

(gdb) c

Continuing.

15:57:07.451 43634 E channel.c:160 | alloc channel 4194336 bytes failed(0)

15:57:07.452 43634 E main.c:276 | context init failed(12)

[Inferior 1 (process 43634) exited with code 01]

(gdb) Killed

让程序正常退出，如果想脱离GDB，请使用detach命令，用q会kill进程，功亏一篑。

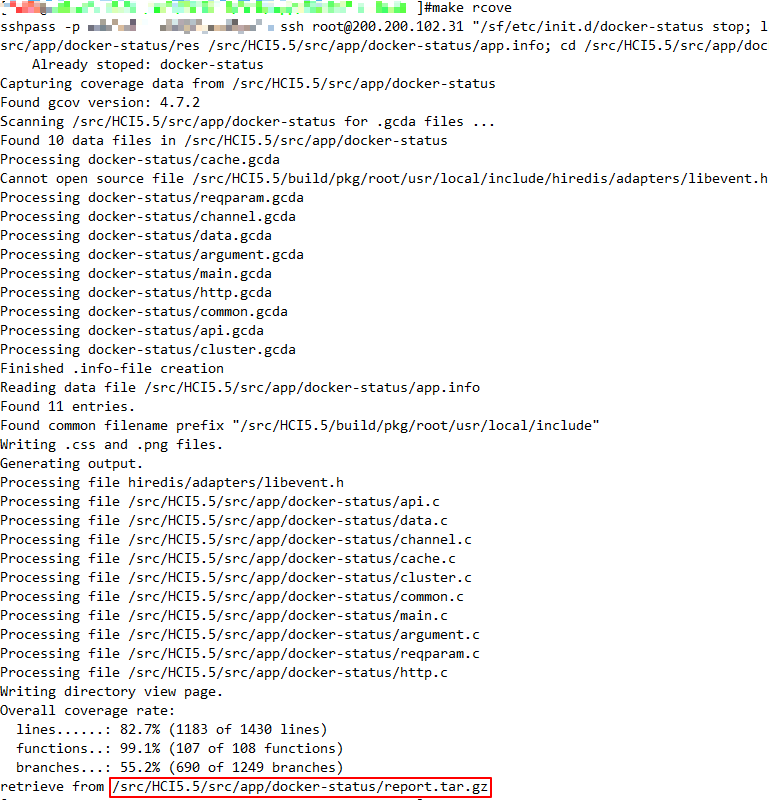
(gdb) detach

运行的时候，切记一点，让程序正常退出，通俗点讲，就是atexit里面的函数也要执行到，不能中途abort或者被KILL。

所以，编码的时候，可以通过捕获SIGTERM来主动退出。

## 收集结果(编译环境)

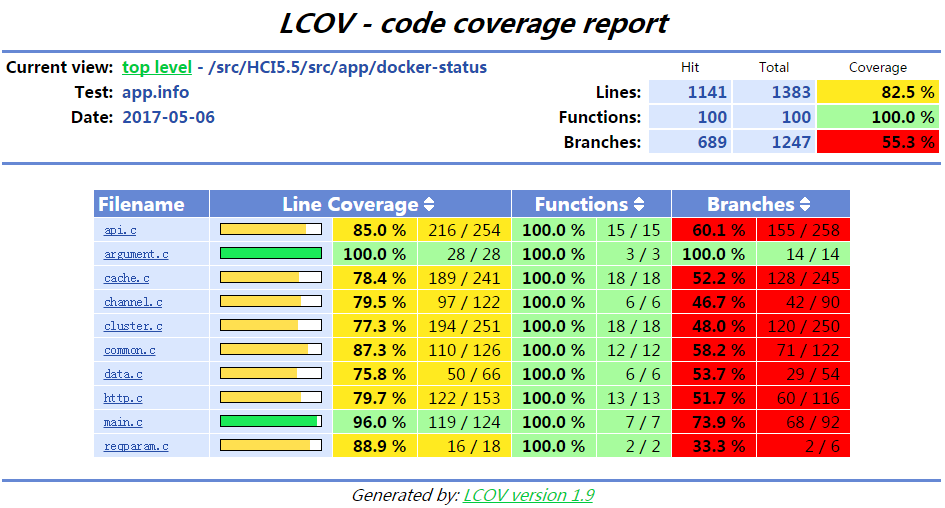
运行完成后，让运行时的程序正常结束，在编译环境上，执行make rcove，完成代码覆盖结果收集。



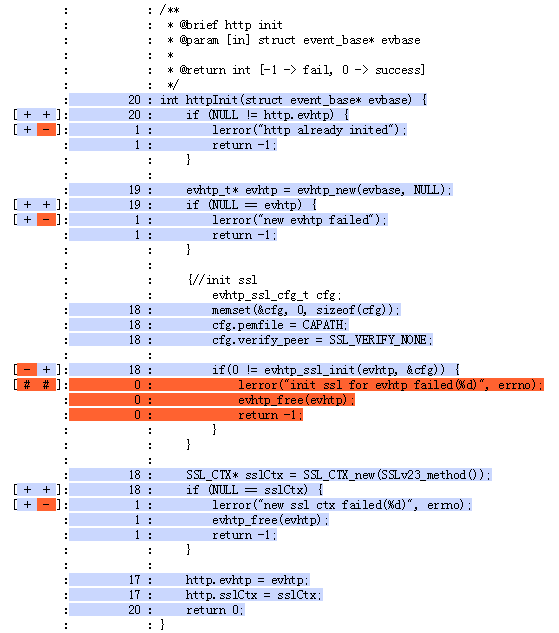
收集到的结果以report.tar.gz方式命令，并放到运行时上述打印目录，sz到本地解压开可以看到index.html。

## 结果分析

如下代码覆盖率：行覆盖82.5%，函数覆盖100%，分支覆盖55.3%



点开可以查看细节：



2017/5/6