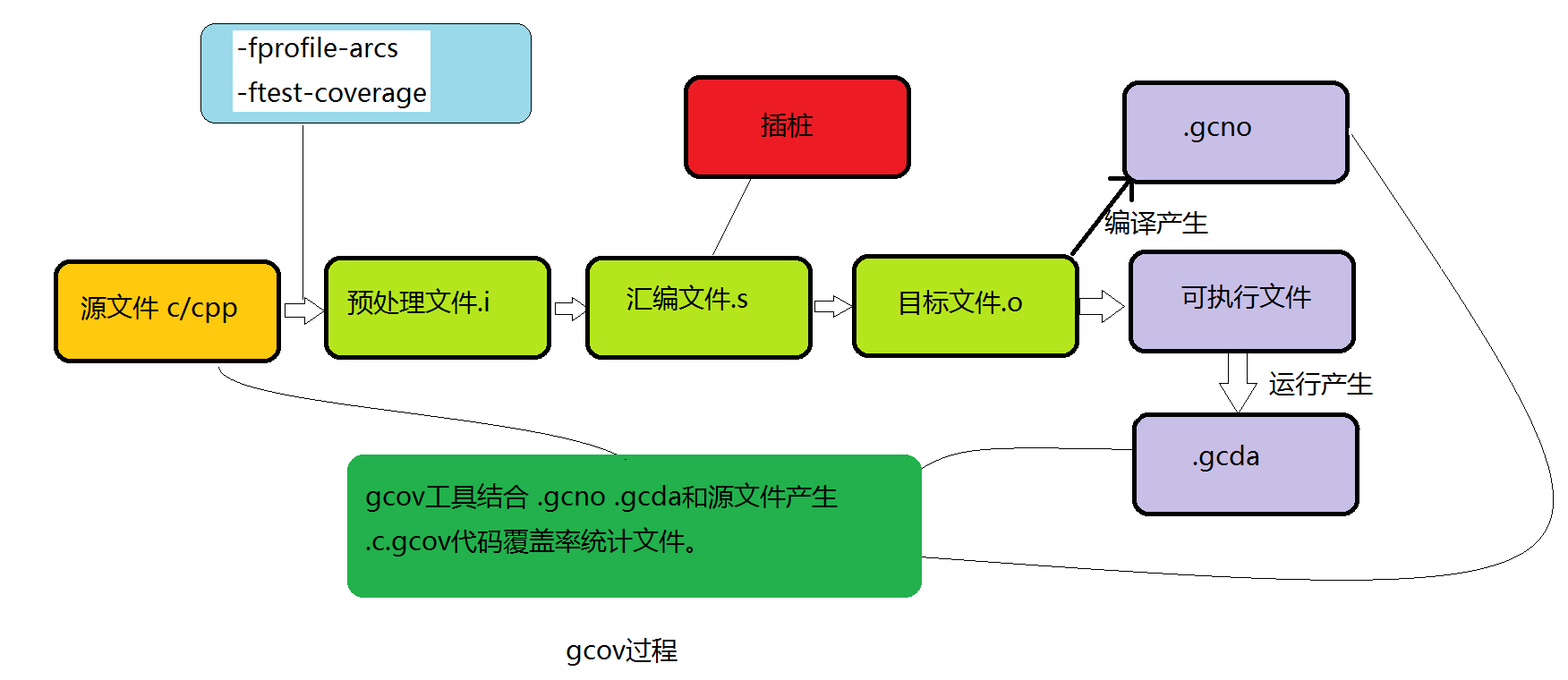
## 1 GCOV介绍

GCOV是随GCC一起发布的用于代码覆盖率统计的工具，一般配合其图形化工具LCOV一起使用。

伴随GCC发布，不需要单独下载gcov工具，需要结合lcov,gcovr等前端图形工具才能实现统计数据图形化。

## 2 gcov过程概况



<主要工作流>

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，然后编译成汇编文件，在生成汇编文件的同时完成插桩。插桩是在生成汇编文件的阶段完成的，因此插桩是汇编时候的插桩，每个桩点插入3~4条汇编语句，直接插入生成的\*.s文件中，最后汇编文件汇编生成目标文件，生成可执行文件；并且生成关联BB和ARC的.gcno文件；

3) 执行可执行文件，在运行过程中之前插入桩点负责收集程序的执行信息。所谓桩点，其实就是一个变量，内存中的一个格子，对应的代码执行一次，则其值增加一次；

4) 生成.gcda文件，其中有BB和ARC的执行统计次数等，由此经过加工可得到覆盖率。

## 3 LCOV介绍

图形化的代码覆盖率统计工具

## 4 linux下安装LCOV

1 在这里下载源代码：<http://sourceforge.net/projects/ltp/>

2 解压缩，进入解压缩后目录

3 打开Makefile文件，将其中的PREFIX：=/usr/local该为PREFIX：=/usr

4保存退出后，执行make install,安装lcov

5 make uninstall 卸载安装

## 5 使用

用GCC编译的时候加上-fprofile-arcs -ftest-coverage选项，链接的时候也加上。

gcov主要使用.gcno和.gcda两个文件。

.gcno是由-ftest-coverage产生的，它包含了重建基本块图和相应的块的源码的行号的信息。

.gcda是由加了-fprofile-arcs编译参数的编译后的文件运行所产生的，它包含了弧跳变的次数和其他的概要信息（而gcda只能在程序运行完毕后才能产生的）。

例子：

生成覆盖率

# gcc -fprofile-arcs -ftest-coverage hello.c -o hello (编译时候加两个参数：-fprofile-arcs -ftest-coverage, 此时会生成：hello.gcno)

# ./hello (此时会生成： hello.gcda, hello.gcno和hello.gcda是gcov引用的关键文件)

# gcov hello.c (gcov根据上面的文件生成了：hello.c.gcov,就是代码覆盖信息，但看起来并不直观)

# lcov -d . -t 'Hello test' -o 'hello\_test.info' -b . -c (我们借助lcov对hello.c.gcov进行改造，可以看见生成了hello\_test.info)

lcov --directory . --capture --output-file app.info

其中：

　　--directory 或者-d 表示的是目录，也就是gcno和gcda目录

　　--capture 或者 -c 表示获取覆盖率信息

　　--output-file 或者 -o 表示输出文件

# genhtml -o result hello\_test.info (生成了result文件夹，借助web服务器，我们就可以很直观的看到结果了)

lcov文件过滤

　a)移除指定目录

lcov --remove all.info '/src/include/\*' '/user/bin/\*' -o result.info

　此命令表示生成的覆盖率信息，屏蔽 '/src/include/\*' '/user/bin/\*' 两个目录的覆盖率信息

　　 all.info 总的覆盖率信息

　 　result.info 筛选后的覆盖率信息

b)只要固定目录

lcov --extract all.info '\*/src/\*' '\*/lib/\*' -o result.info

此命令表示生成的覆盖率信息只要src和lib目录下文件的覆盖率信息

　　all.info 总的覆盖率信息

　result.info 筛选后的覆盖率信息

## 6 编写makefile，生成覆盖率文件

例子：

|  |
| --- |
| CC=gcc  CFLAGS=-I.  DEPS= hello.h  OBJ = hello.o hellofunc.o  COV = -ftest-coverage -fprofile-arcs  %.o: %.c $(DEPS)  $(CC) -c -o $@ $< $(CFLAGS) $(COV)    hello: $(OBJ)  gcc -o $@ $^ $(CFLAGS) --coverage  ./hello  lcov -c -o $@.info -d .  genhtml $@.info -o $@\_result  open $@\_result/index.html    clean:  rm -f $(OBJ) hello hello.info hello.g\* hellofunc.g\*  rm -rf hello\_result |