Code Coverage Analysis using gcov

Author : Vikas Nagpal (https://github.com/vikasnagpaliitd)
Version 1.0

Introduction

How do we know we have tested out code well?

One measure is that we have executed each code statement at least once.

Code Coverage tools, like gcov, help us know which lines of the code have been executed (and how many times)

Quality metrics and customer agreements often demand close to 100% code coverage reports.

Compiling code for gcov

$ gcc **-fprofile-arcs -ftest-coverage** <file1.c> <file2.c> …

Please note that **'gcov notes' files** get created after the above command. e.g. file1.gcno, file2.gcno, etc. These notes files contain structural information about the code, which will help 'gcov' understand the 'gcov data' files created during execution of the executable.

Also, the executable has got added instrumentation code, to keep measuring code coverage.

When we run the instrumented executable, **"gcov data" files** are created or updated. e.g. file1.gcda, file2.gcda, etc. These contain cumulative information about how many times different source code lines are executed.

Coverage Analysis of project1

project1 comprises of single main.c file

```c
#include <stdio.h>

int main()
{
    int x;

    printf("Enter an integer: ");
    scanf("%d", &x);

    if (x < 10)
        printf("x is less than 10\n");
    else if (x < 20)
        printf("x is greater than or equal to 10 and x is less than 20\n");
    else
        printf("x is greater than or equal to 20\n");

    return 0;
}
```

Let us compile it for gcov

```
anuttara:project1>gcc -fprofile-arcs -ftest-coverage main.c
anuttara:project1>ls
a.out  main.c  main.gcno
anuttara:project1>
```

Let us now run a test case with input integer value being 1.

```
anuttara:project1>./a.out
Enter an integer: 1
x is less than 10
anuttara:project1>ls
a.out  main.c  main.gcda  main.gcno
anuttara:project1># Note above the formation of main.gcda
anuttara:project1># Below command generates code coverage report
anuttara:project1>gcov main.c
File 'main.c'
Lines executed:66.67% of 9
Creating 'main.c.gcov'
```

```
anuttara:project1>cat main.c.gcov
        -:    0:Source:main.c
        -:    0:Graph:main.gcno
        -:    0:Data:main.gcda
        -:    0:Runs:1
        -:    1:#include <stdio.h>
        -:    2:
        1:    3:int main()
        -:    4:{
        -:    5:    int x;
        -:    6:
        1:    7:    printf("Enter an integer: ");
        1:    8:    scanf("%d", &x);
        -:    9:
        1:   10:    if (x < 10)
        1:   11:        printf("x is less than 10\n");
    #####:   12:    else if (x < 20)
    #####:   13:        printf("x is greater than or equal to 10 and x is
less than 20\n");
        -:   14:    else
    #####:   15:        printf("x is greater than or equal to 20\n");
        -:   16:
        1:   17:    return 0;
        -:   18:}
anuttara:project1>
```

The first column indicates how many times a line is executed.It might have

- a number : indicates that the line is executed that many times
- - : indicates that the line is not an executable statement
- ##### : indicates that the line is not executed yet. More test cases needs to be added

The "####" are clearly indicating that we need to run test cases to reach line#12, 13 and 15
Let us do that. We will run a.out with input 15 and 25.

```
anuttara:project1>./a.out
Enter an integer: 15
x is greater than or equal to 10 and x is less than 20
anuttara:project1>
anuttara:project1>gcov main.c
File 'main.c'
Lines executed:88.89% of 9
```

```
Creating 'main.c.gcov'

anuttara:project1># Notice : coverage percentage has increased
anuttara:project1>
anuttara:project1>cat main.c.gcov
        -:    0:Source:main.c
        -:    0:Graph:main.gcno
        -:    0:Data:main.gcda
        -:    0:Runs:2
        -:    1:#include <stdio.h>
        -:    2:
        2:    3:int main()
        -:    4:{
        -:    5:    int x;
        -:    6:
        2:    7:    printf("Enter an integer: ");
        2:    8:    scanf("%d", &x);
        -:    9:
        2:   10:    if (x < 10)
        1:   11:        printf("x is less than 10\n");
        1:   12:    else if (x < 20)
        1:   13:        printf("x is greater than or equal to 10 and x is
less than 20\n");
        -:   14:    else
    #####:   15:        printf("x is greater than or equal to 20\n");
        -:   16:
        2:   17:    return 0;
        -:   18:}
anuttara:project1># Notice : just line 15 remains to be executed
anuttara:project1>
anuttara:project1># Let us run the code with value 25
anuttara:project1>./a.out
Enter an integer: 25
x is greater than or equal to 20
anuttara:project1>
anuttara:project1>gcov main.c
File 'main.c'
Lines executed:100.00% of 9
Creating 'main.c.gcov'

anuttara:project1>
anuttara:project1># We achieved 100% code coverage
anuttara:project1>
```

```
anuttara:project1>gcov main.c
File 'main.c'
Lines executed:100.00% of 9
Creating 'main.c.gcov'

anuttara:project1>
anuttara:project1>cat main.c.gcov
        -:    0:Source:main.c
        -:    0:Graph:main.gcno
        -:    0:Data:main.gcda
        -:    0:Runs:3
        -:    1:#include <stdio.h>
        -:    2:
        3:    3:int main()
        -:    4:{
        -:    5:    int x;
        -:    6:
        3:    7:    printf("Enter an integer: ");
        3:    8:    scanf("%d", &x);
        -:    9:
        3:   10:    if (x < 10)
        1:   11:        printf("x is less than 10\n");
        2:   12:    else if (x < 20)
        1:   13:        printf("x is greater than or equal to 10 and x is
less than 20\n");
        -:   14:    else
        1:   15:        printf("x is greater than or equal to 20\n");
        -:   16:
        3:   17:    return 0;
        -:   18:}
anuttara:project1>
anuttara:project1># There is no line with "#####".
anuttara:project1># Also note, that some lines executed 3 times
```

Notes

1. Do not use optimization options while using code coverage. That changes source code line number to machine code mapping adversely
2. Running the executable just updates “.gcda” files. The “.c.gcov” has to be generated manually for each source code file by doing “gcov <file.c>”. This command correlates the

structural information in “.gcno” file with data information in “.gcda” file, to generate textual report in “.c.gcov” file

References

1) gcov : a Test Coverage Program
2) An Introduction to GCC - 10.3 Coverage testing with gcov
3) Gcov Intro (Using the GNU Compiler Collection (GCC))
4) Youtube Video : https://youtu.be/ZvaurxqdOnA