GNU `gcov` (1/6) [from Wikipedia]

- `gcov` is a source code coverage analysis and statement-by-statement profiling tool.
- `gcov` generates exact counts of the number of times each statement in a program has been executed
- `gcov` does not produce any time-based data (you should use `gprof` for this purpose) and works only on code compiled with the GCC suite.

GNU `gcov` (2/6)

- To use gcov, each source file should be compiled with `--coverage`, which generates a `.gcno` file that is a graph file of the source file.
- After the instrumented target program completes its execution, execution statistics is recorded in a `.gcda` file.
- gcov creates a human readable logfile `.gcov` from a binary `.gcda` file, which indicates how many times each line of a source file has executed.
- **gcov [-b] [-c] [-v] [-n] [-l] [-f] [-o directory] *sourcefile***
  - -a: Write individual execution counts for every basic block.
  - -b: Write branch frequencies to the output file
  - -c: Write branch frequencies as the number of branches taken
  - -f: Output summaries for each function in addition to the file level summary.
  - -o The directory where the object files live. Gcov will search for `.bb', `.bbg', and `.da' files in this directory

GNU gcov (3/6)

- For example, if you measure coverage of example.c,

```
[moonzoo@verifier gcov]$ l
example.c
[moonzoo@verifier gcov]$ gcc -fprofile-arcs
      -ftest-coverage example.c
[moonzoo@verifier gcov]$ a.out 5
i=5
j=2
[moonzoo@verifier gcov]$ gcov -b example.c
File 'example.c'
Lines executed:78.57% of 14
Branches executed:100.00% of 10
Taken at least once:50.00% of 10
Calls executed:60.00% of 5
example.c:creating 'example.c.gcov'
```

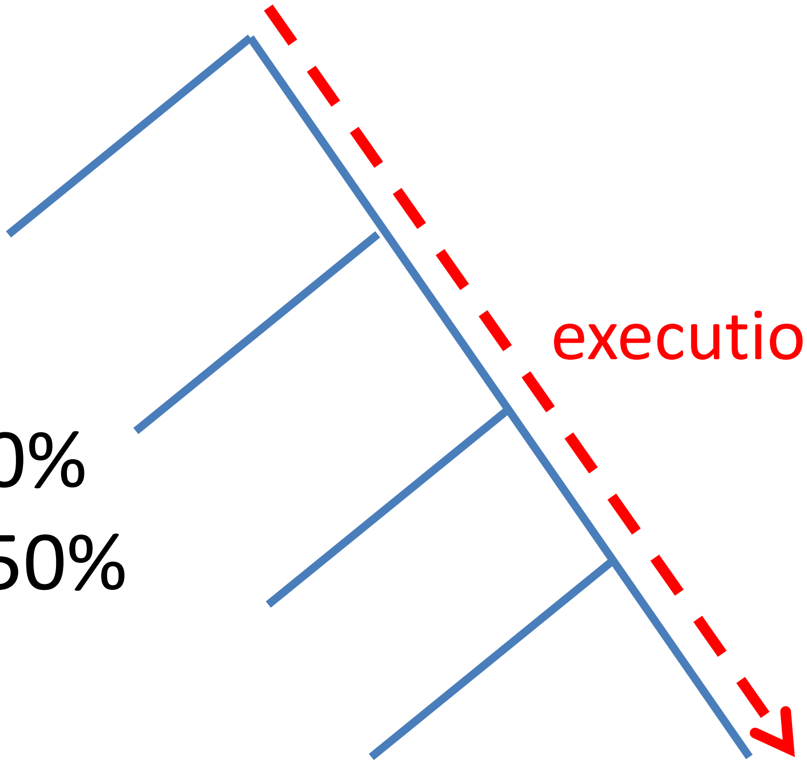
```c
#include <stdio.h>
int main(int argc, char **argv){
    int i=0,j=0;
    if (argc < 2) {
        printf("Usage:...₩n”);exit(-1);}
    i = atoi(argv[1]);
    printf("i=%d₩n",i);

    if( i == 0)
        j=0;
     else {
        if (i == 1)
            j=1;
        if (i > 1 && i < 10)
            j=2;
     }
     printf("j=%d₩n",j);
}
```

GNU `gcov` (4/6)

“Branches executed” vs. “Taken at least once”

- For measuring branch coverage, be careful to use “Taken at least once”, not “Branches executed”

Ex.
Branch executed: 100%
Taken at least once: 50%



GNU gcov (5/6)

```
1 #include <stdio.h>
2 int main(int argc, char **argv){
3      int i=0,j=0;
4      if (argc < 2) {
5           printf("Usage:...₩n");exit(-1);}
6      i = atoi(argv[1]);
7      printf("i=%d₩n",i);
8
9      if( i == 0)
10          j=0;
11      else {
12          if (i == 1)
13              j=1;
14          if (i > 1 && i < 10)
15              j=2;
16      }
17      printf("j=%d₩n",j);
18 }
```

Executed function info

Not executed

Call info

Non-executable statement

Branch info for each condition

```
        -:    0:Source:example.c
        -:    0:Graph:example.gcno
        -:    0:Data:example.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
function main called 1 returned
100% blocks executed 71%
        1:    2:int main(int argc,
char **argv){
        1:    3:    int i=0,j=0;
        1:    4:    if (argc < 2) {
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
    #####:    5:
printf("Usage:…\n");exit(-1);}
call    0 never executed
call    1 never executed
        1:    6:    i=atoi(argv[1]);
call    0 returned 100%
        1:    7:
printf("i=%d\n",i);
call    0 returned 100%
        -:    8:
        1:    9:    if( i == 0)
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
    #####:   10:        j=0;
        -:   11:    else {
        1:   12:        if (i == 1)
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
    #####:   13:            j=1;
        1:   14:        if(i>1&&i<10)
branch  0 taken 100% (fallthrough)
branch  1 taken 0%
branch  2 taken 100% (fallthrough)
branch  3 taken 0%
        1:   15:            j=2;
        -:   16:    }
        1:   17:
printf("j=%d\n",j);
call    0 returned 100%
        1:   18:}
```

<u>Note that a "branch" for gcov is anything that causes the code to execute non-straight line</u>

Conditional statement with a compound condition (i.e., a Boolean formula containing && or ||) has more than 2 branches

GNU gcov (6/6)

```
 1 #include <stdio.h>
 2 int main(int argc, char **argv){
 3      int i=0,j=0;
 4      if (argc < 2) {
 5          printf("Usage:...₩n");exit(-1);}
 6      i = atoi(argv[1]);
 7      printf("i=%d₩n",i);
 8
 9      if( i == 0)
10          j=0;
11      else {
12          if (i == 1)
13              j=1;
14          if (i > 1 && i < 10)
15              j=2;
16      }
17      printf("j=%d₩n",j);
18 }
```

This intermediate coverage information format (obtained with –i option) is mainly used by coverage visualization tools like lcov, gcovr, etc.

Note that recent `gcov -i` (version 9.1 or higher) generates the intermediate coverage information in a json file.

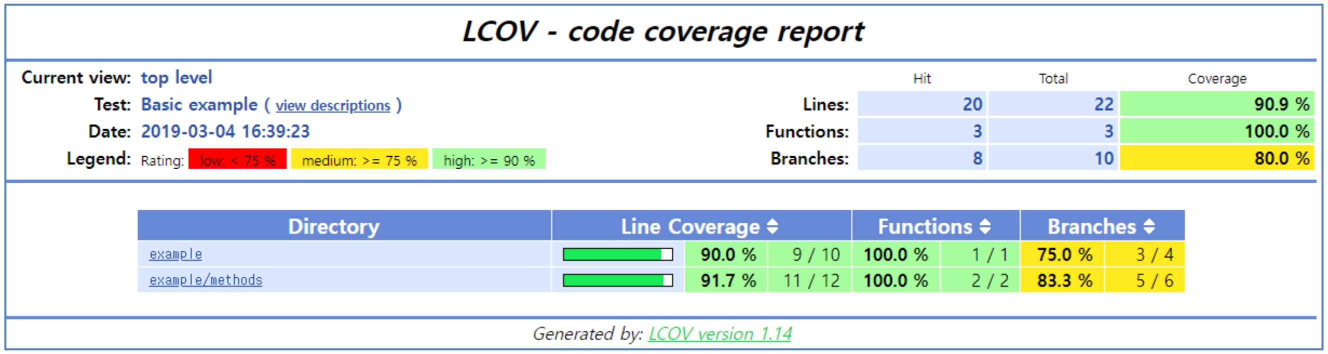
Intermediate output

```
mz@hp-x360-mz:~/tmp$ ./a.out 5
i=5
j=2

mz@hp-x360-mz:~/tmp$ gcov -b -i example.c
File 'example.c'
Lines executed:84.62% of 13
Branches executed:100.00% of 10
Taken at least once:60.00% of 10
Calls executed:100.00% of 5
Creating 'example.c.gcov'

mz@hp-x360-mz:~/tmp$ cat example.c.gcov
file:example.c
function:2,1,main
lcount:2,1
lcount:3,1
lcount:4,1
branch:4,nottaken
lcount:5,0
lcount:6,1
lcount:7,1
lcount:9,1
branch:9,nottaken
lcount:10,0
lcount:12,1
branch:12,nottaken
lcount:13,0
lcount:14,1
branch:14,taken
branch:14,taken
lcount:15,1
lcount:17,1
```

Visual Coverage Tools based on `gcov`

- `lcov`
  - https://github.com/linux-test-project/lcov
- `gcovr`
  - https://github.com/linux-test-project/lcov
- `grcov`
  - https://github.com/mozilla/grcov

lcov



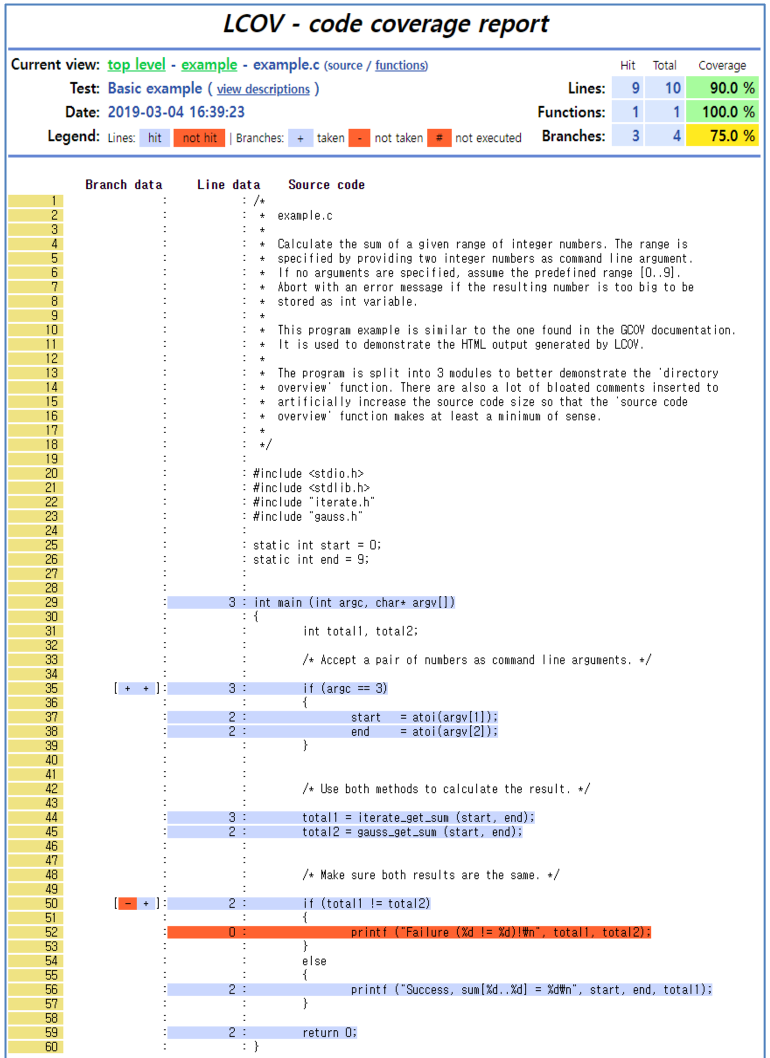
LCOV - code coverage report

Current view: top level
Test: Basic example (view descriptions)
Date: 2019-03-04 16:39:23
Legend: Rating: low: < 75 % medium: >= 75 % high: >= 90 %

	Hit	Total	Coverage
Lines:	20	22	90.9 %
Functions:	3	3	100.0 %
Branches:	8	10	80.0 %

Directory	Line Coverage			Functions		Branches	
example		90.0 %	9 / 10	100.0 %	1 / 1	75.0 %	3 / 4
example/methods		91.7 %	11 / 12	100.0 %	2 / 2	83.3 %	5 / 6

Generated by: LCOV version 1.14



LCOV - code coverage report

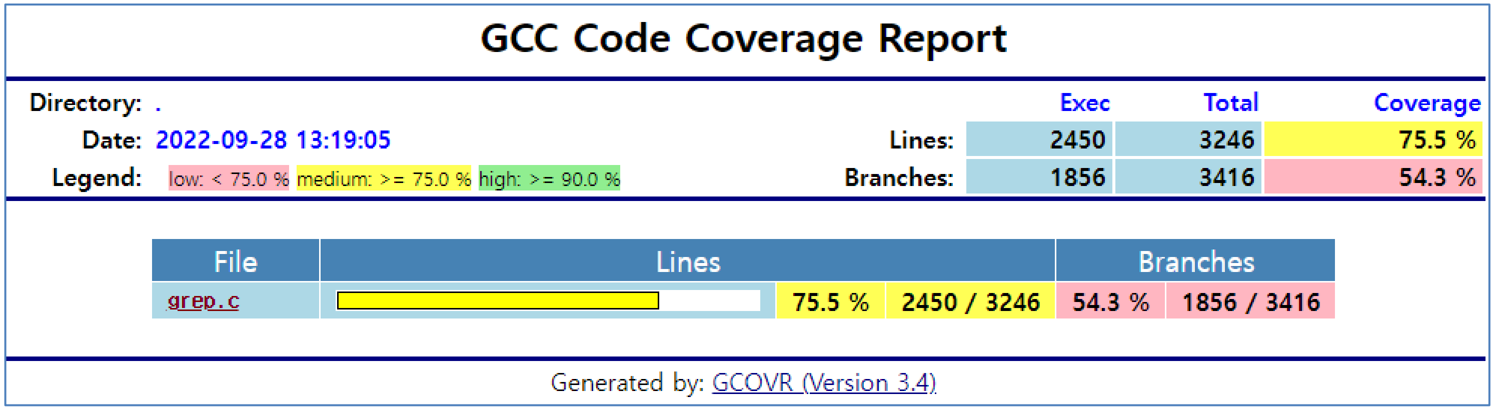
Current view: top level - example - example.c (source / functions)
Test: Basic example (view descriptions)
Date: 2019-03-04 16:39:23
Legend: Lines: hit not hit | Branches: + taken - not taken # not executed

	Hit	Total	Coverage
Lines:	9	10	90.0 %
Functions:	1	1	100.0 %
Branches:	3	4	75.0 %

```
     Branch data     Line data    Source code
                :              : /*
                :              :  *  example.c
                :              :  *
                :              :  *  Calculate the sum of a given range of integer numbers. The range is
                :              :  *  specified by providing two integer numbers as command line argument.
                :              :  *  If no arguments are specified, assume the predefined range [0..9].
                :              :  *  Abort with an error message if the resulting number is too big to be
                :              :  *  stored as int variable.
                :              :  *
                :              :  *  This program example is similar to the one found in the GCOV documentation.
                :              :  *  It is used to demonstrate the HTML output generated by LCOV.
                :              :  *
                :              :  *  The program is split into 3 modules to better demonstrate the 'directory
                :              :  *  overview' function. There are also a lot of bloated comments inserted to
                :              :  *  artificially increase the source code size so that the 'source code
                :              :  *  overview' function makes at least a minimum of sense.
                :              :  *
                :              :  */
                :              : 
                :              : #include <stdio.h>
                :              : #include <stdlib.h>
                :              : #include "iterate.h"
                :              : #include "gauss.h"
                :              : 
                :              : static int start = 0;
                :              : static int end = 9;
                :              : 
                :              : 
                :            3 : int main (int argc, char* argv[])
                :              : {
                :              :         int total1, total2;
                :              : 
                :              :         /* Accept a pair of numbers as command line arguments. */
                :              : 
          [ + + ]:            3 :         if (argc == 3)
                :              :         {
                :            2 :                 start   = atoi(argv[1]);
                :            2 :                 end     = atoi(argv[2]);
                :              :         }
                :              : 
                :              : 
                :              :         /* Use both methods to calculate the result. */
                :              : 
                :            3 :         total1 = iterate_get_sum (start, end);
                :            2 :         total2 = gauss_get_sum (start, end);
                :              : 
                :              : 
                :              :         /* Make sure both results are the same. */
                :              : 
          [ - + ]:            2 :         if (total1 != total2)
                :              :         {
                :            0 :                 printf ("Failure (%d != %d)!₩n", total1, total2);
                :              :         }
                :              :         else
                :              :         {
                :            2 :                 printf ("Success, sum[%d..%d] = %d₩n", start, end, total1);
                :              :         }
                :              : 
                :            2 :         return 0;
                :              : }
```

gcovr

```
grep-v2.0-simplified$ gcovr --html --html-detail -o index.html
```

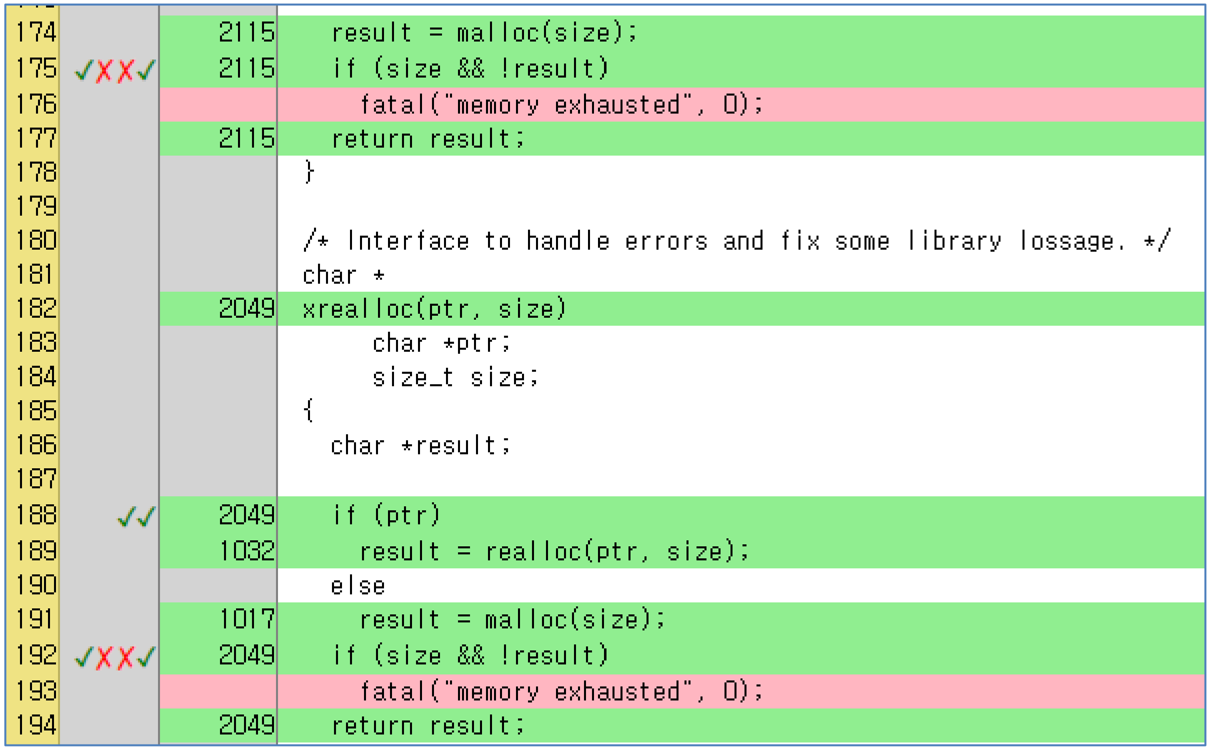


GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	.				
Date:	2022-09-28 13:19:05	Lines:	2450	3246	75.5 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	Branches:	1856	3416	54.3 %

File	Lines			Branches	
grep.c		75.5 %	2450 / 3246	54.3 %	1856 / 3416

Generated by: GCOVR (Version 3.4)



```
             2115      result = malloc(size);
   ✓XX✓      2115      if (size && !result)
                         fatal("memory exhausted", 0);
             2115      return result;
                     }

                     /* Interface to handle errors and fix some library lossage. */
                     char *
             2049    xrealloc(ptr, size)
                          char *ptr;
                          size_t size;
                     {
                       char *result;

     ✓✓      2049      if (ptr)
             1032        result = realloc(ptr, size);
                       else
             1017        result = malloc(size);
   ✓XX✓      2049      if (size && !result)
                         fatal("memory exhausted", 0);
             2049      return result;
```

gcovr

```
grep-v2.0-simplified$ gcovr -b
------------------------------------------------------------------------------
                           GCC Code Coverage Report
Directory: .
------------------------------------------------------------------------------
File                                       Branches    Taken  Cover    Missing
------------------------------------------------------------------------------
grep.c                                          3416      1856     54%
147,175,192,252,259,284,287,289,293,311,312,317,402,419,457,508,552,579,588,643,657,679,686,691,696,704,725,72
6,741,789,800,810,813,821,825,835,838,846,848,851,854,1712,2782,2806,2808,2818,2828,2839,2840,2854,2855,2865,
2870,2872,2874,2898,2900,2901,2904,2906,2908,2909,2929,2948,2955,2956,2957,2958,2973,2974,2987,2997,3005,30
09,3015,3023,3030,3036,3038,3041,3043,3045,3053,3058,3065,3066,3067,3068,3072,3075,3080,3083,3089,3093,3097
,3101,3102,3103,3112,3128,3132,3134,3140,3157,3173,3195,3209,3216,3221,3233,3237,3256,3259,3277,3278,3284,3
289,3297,3298,3327,3333,3341,3344,3349,3377,3387,3390,3391,3403,3405,3411,3415,3422,3423,3425,3433,3435,343
8,3440,3449,3451,3453,3462,3464,3481,3483,3490,3502,3545,3547,3549,3577,3583,3588,3592,3596,3600,3604,3608,
3613,3618,3622,3626,3642,3658,3661,3663,3664,3665,3666,3668,3674,3678,3690,3787,3791,3793,3807,3812,3814,38
15,3829,3832,3862,3880,3881,3889,4211,4216,4218,4234,4257,4258,4265,4268,4269,4275,4276,4282,4283,4290,4294
,4299,4355,4363,4364,4372,4373,4391,4393,4399,4414,4474,4544,4549,4551,4556,4558,4565,4566,4576,4578,4584,4
587,4591,4592,4593,4597,4604,4608,4614,4615,4620,4623,4627,4629,4917,4924,4936,4937,4955,4977,4988,5005,503
9,5041,5046,5047,5064,5075,5076,5078,5089,5099,5103,5107,5112,5126,5129,5137,5141,5142,5161,5174,5193,5197,
5198,5201,5207,5220,5222,5223,5226,5240,5241,5253,5270,5280,5281,5296,5316,5317,5329,5340,5350,5364,5365,53
66,5372,5381,5395,5396,5408,5415,5420,5426,5446,5457,5458,5459,5464,5466,5467,5474,5477,5484,5489,5505,5507
,5509,5521,5523,5527,5528,5538,5546,5573,5611,5614,5621,5626,5652,5653,5654,5658,5669,5671,5678,5685,5686,5
690,5691,5696,5732,5765,5778,5796,5835,5841,5847,5853,5854,5918,5919,5927,5928,5930,5943,5956,5962,5973,597
4,5975,5983,5990,6020,6026,6036,6056,6062,6072,6079,6092,6107,6128,6133,6143,6167,6193,6196,6203,6214,6225,
6252,6254,6399,6412,6417,6421,6422,6428,6446,6478,6490,6495,6500,6505,6507,6511,6541,6542,6553,6558,6560,65
79,6586,6591,6718,6730,6743,6974,6993,7001,7007,7012,7017,7019,7021,7022,7023,7024,7025,7026,7027,7028,7041
,7069,7073,7078,7080,7085,7089,7094,7104,7105,7110,7111,7116,7121,7130,7132,7134,7140,7148,7149,7150,7165,7
172,7176,7178,7197,7200,7203,7216,7217,7225,7228,7229,7230,7233,7234,7244,7245,7246,7247,7269,7388,7402,744
2,7508,7516,7522,7584,7586,7607,7610,7643,7645,7647,7665,7666,7667,7668,7670,7710,7716,7876,7924,8148,8210,
8240,8255,8297,8327,8329,8330,8335,8347,8350,8353,8475,8479,8493,8507,8551,8590,8593,8594,8597,8598,8600,86
05,8701,8711,8742,8752,8769,8771,8799,8819,8822,8831,8839,8852,8857,8874,8877,8885,8892,8932,8943,8944,8958
,8960,8977,8984,9009,9011,9015,9023,9029,9038,9050,9058,9061,9066,9074,9081,9087,9095,9100,9117,9286,9292,9
293,9362,9364,9368,9370,9402,9405,9423,9434,9587,9591,9599,9704,9706,9726,9731,9743,9745,9747,9760,9762,976
4,9766,9794,9801,9833,9839,9841,9843,9844,9861,9874,9876,9879,9884,9890,9892,9894,9895,9899,9902,9912,9917,
9921,9948,9951,9971,10577,10602,10611,10614,10632,10687,10698,10773,10775,10782,10806,10809,10817,10818,1
0820,10821,10823,10830,10833,10866,10874,10888,10893,10895,10897,10901,10902,10904,10906,10921
------------------------------------------------------------------------------
TOTAL                                             3416      1856     54%
------------------------------------------------------------------------------
```

clang, llvm-cov

You can use clang/llvm-cov tools in a similar way to gcc/gcov.

1. To build a coverage-enabled executable, run

```
clang --coverage example.c -o example
```

2. Run generated executable

```
./example
```

3. To get coverage of example.c, run

```
llvm-cov gcov -b example.c
```

4. Then, you will get the same result as `gcov` (including the same `example.c.gcov` file)

```
File 'example.c'
Lines executed:28.57% of 14
Branches executed:20.00% of 10
Taken at least once:10.00% of 10
No calls
Creating 'example.c.gcov'
```