GNU `gcov` (1/4) [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 (2/4)

- To use gcov, each source file should be compiled with `-fprofile-arcs` and `-ftest-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 (3/4)

- 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'
```

```
 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 }
```

GNU gcov (4/4)

```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);
}
```

Executed function info

Not executed

Call info

Non-executable statement

Branch info for each condition

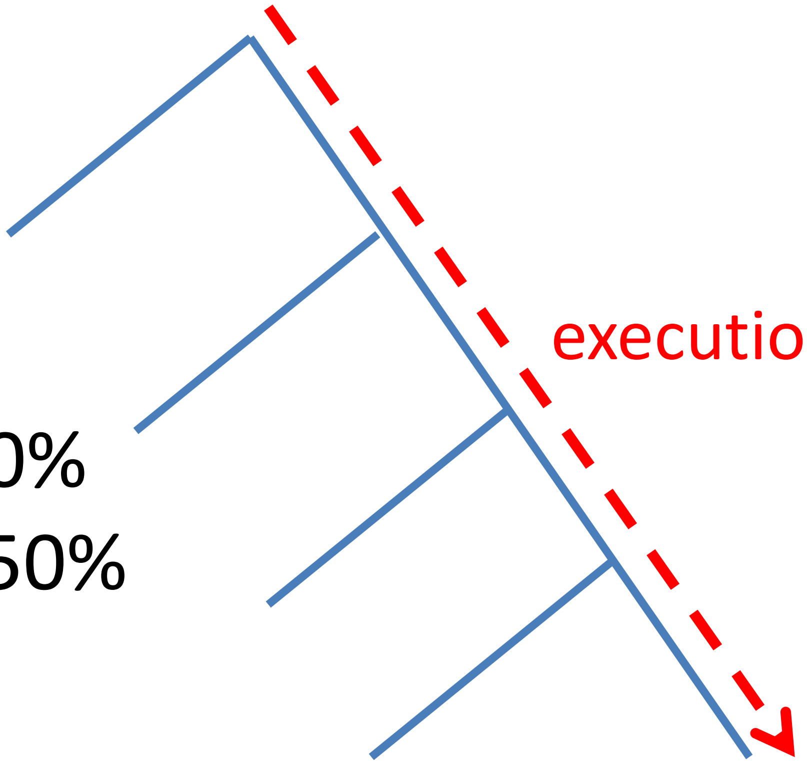
```
        -:    0:Source:example.c
        -:    0:Graph:example.gcno
        -:    0:Data:example.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
function main called 1 returned
100% blocks executed 71%
        1:    2:int main(int argc,
char **argv){
        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

"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%