Inspecting the Assembly output of -fprofile-arcs.

To understand code coverage for GlibC in particular, it was suggested that looking at the assembly output of a small code with the -fprofile-arcs might help. I'm just documenting all my findings below. I don't yet know what they all mean, but hopefully I can work it out.

Step 1: Small program

- On the lines of the toupper program, I wrote a very small program to take an initialised char and convert it to uppercase.

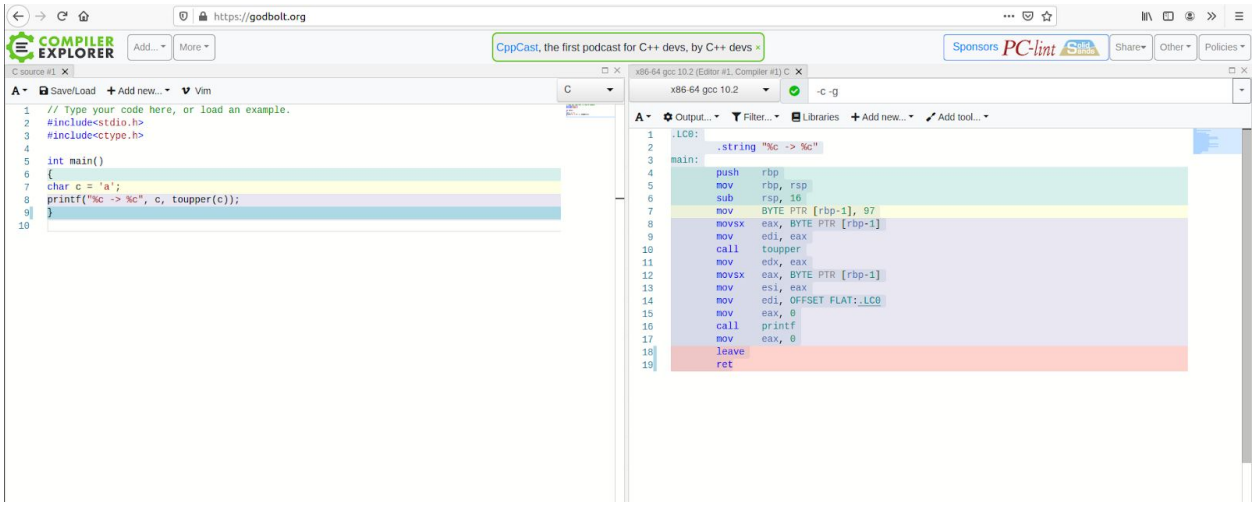
Code:

```
#include<stdio.h>
#include<ctype.h>

int main()
{
char c = 'a';
printf("%c -> %c", c, toupper(c));
}
```

Output: a -> A

ASM output:

Using https://godbolt.org/ with language as C and compiler as **x86-64 gcc 10.2** and no flags, we get the following ASM output:



```
.LC0:
.string "%c -> %c"
main:
push rbp
mov rbp, rsp
sub rsp, 16
mov BYTE PTR [rbp-1], 97
movsx eax, BYTE PTR [rbp-1]
mov edi, eax
call toupper
mov edx, eax
movsx eax, BYTE PTR [rbp-1]
mov esi, eax
mov edi, OFFSET FLAT:.LC0
mov eax, 0
call printf
mov eax, 0
leave
ret
```

I wanted to be sure that my PC was giving the same output godbolt was giving. So I chose the same version of GCC that was on my PC (gcc 9.3.0).

However running:

gcc -S toupper.c

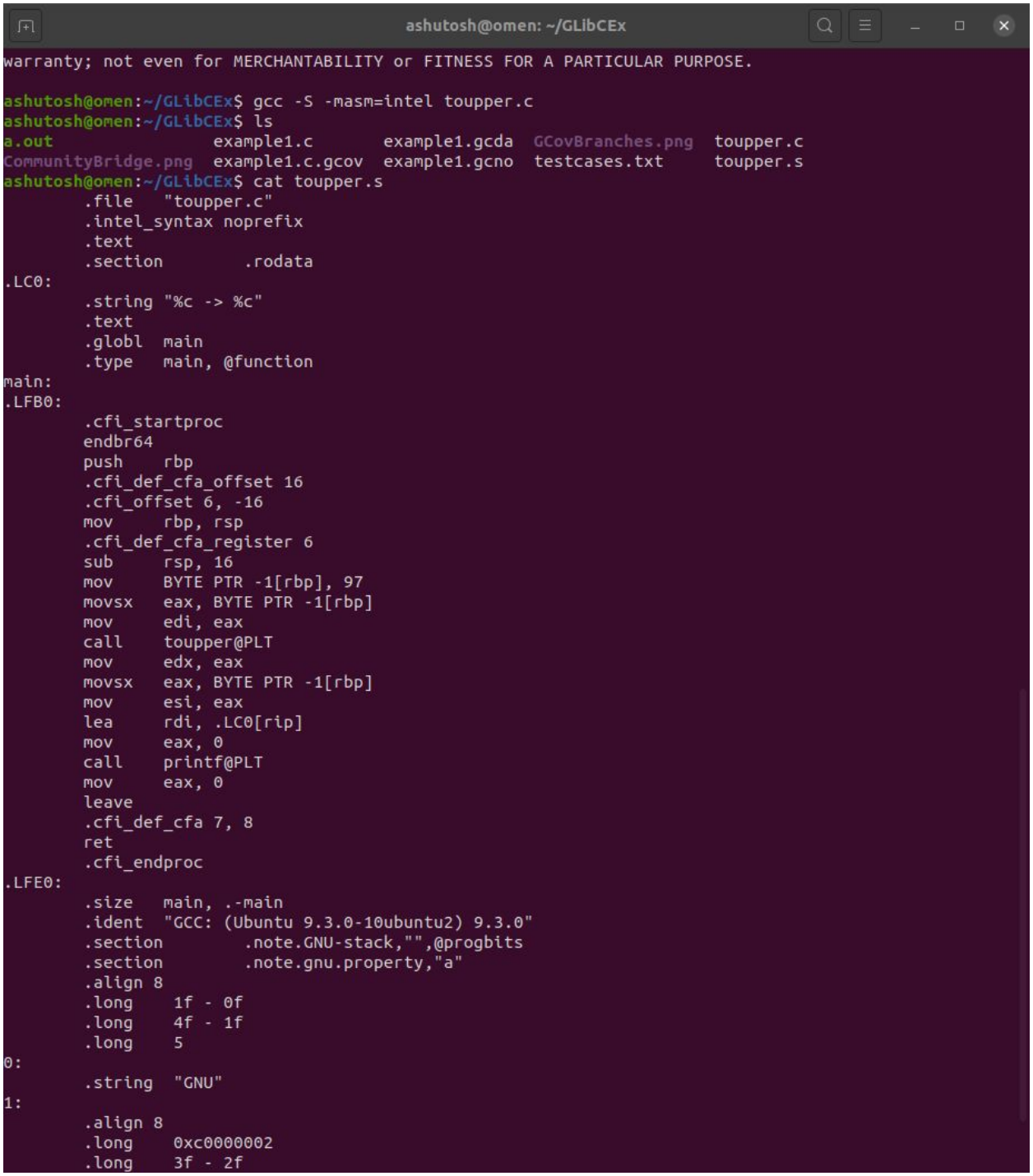
gave a very different looking file, I checked stackoverflow:
https://stackoverflow.com/questions/199966/how-do-you-use-gcc-to-generate-assembly-code-in-intel-syntax

The godbolt compiler explorer gave **intel asm syntax** while the default gcc -S flag gives a different syntax. So I tried:

gcc -S -masm=intel toupper.c

The generated file this time was exactly the same as the one godbolt generated. There were a few differences but all the instructions were the same. This confirmed that I could rely on Godbolt to further inspect assembly code.

The output of gcc -S -masm=intel toupper.c is given below:



-fprofile-arcs and GCDA

As discussed in the meeting, we need to understand how the instrumentation for code coverage works (the format of the GCNO and GCDA files). So we need to know what the compiler exactly does when -fprofile-arcs

and -ftest-coverage flags are added, since the only thing -gcov does is generate a human readable file from the gcda and gcno files.

From page 220 and 221 of the GNU GCC manual https://gcc.gnu.org/onlinedocs/gcc-9.3.0/gcc/

> `-fprofile-arcs`
>
> Add code so that program flow *arcs* are instrumented. During execution the program records how many times each branch and call is executed and how many times it is taken or returns. On targets that support constructors with priority support, profiling properly handles constructors, destructors and C++ constructors (and destructors) of classes which are used as a type of a global variable.
>
> When the compiled program exits it saves this data to a file called '*auxname*`.gcda`' for each source file. The data may be used for profile-directed optimizations ('`-fbranch-probabilities`'), or for test coverage analysis ('`-ftest-coverage`'). Each object file's *auxname* is generated from the name of the output file, if explicitly specified and it is not the final executable, otherwise it is the basename of the source file. In both cases any suffix is removed (e.g. '`foo.gcda`' for input file '`dir/foo.c`', or '`dir/foo.gcda`' for output file specified as '`-o dir/foo.o`'). See Section 10.5 [Cross-profiling], page 887.
>
> [illegible] information.
>
> With '`-fprofile-arcs`', for each function of your program GCC creates a program flow graph, then finds a spanning tree for the graph. Only arcs that are not on the spanning tree have to be instrumented: the compiler adds code to count the number of times that these arcs are executed. When an arc is the only exit or only entrance to a block, the instrumentation code can be added to the block; otherwise, a new basic block must be created to hold the instrumentation code.
>
> `-ftest-coverage`
>
> Produce a notes file that the `gcov` code-coverage utility (see Chapter 10 [`gcov`—a Test Coverage Program], page 875) can use to show program coverage. Each source file's note file is called '*auxname*`.gcno`'. Refer to the '`-fprofile-arcs`' option above for a description of *auxname* and instructions on how to generate test coverage data. Coverage data matches the source files more closely if you do not optimize.

Then I ran

gcc -c -g -fprofile-arcs toupper.c

And got the following output:

```
.LC0:
.string "%c -> %c"
main:
push rbp
mov rbp, rsp
sub rsp, 16
mov rax, QWORD PTR __gcov0.main[rip]
add rax, 1
```

```
mov QWORD PTR __gcov0.main[rip], rax
mov BYTE PTR [rbp-1], 97
movsx eax, BYTE PTR [rbp-1]
mov edi, eax
call toupper
mov edx, eax
movsx eax, BYTE PTR [rbp-1]
mov esi, eax
mov edi, OFFSET FLAT:.LC0
mov eax, 0
call printf
mov rax, QWORD PTR __gcov0.main[rip+8]
add rax, 1
mov QWORD PTR __gcov0.main[rip+8], rax
mov eax, 0
leave
ret
__gcov_.main:
.quad .LPBX0
.long 108032747
.long 1023413812
.long 2092124921
.zero 4
.long 2
.zero 4
.quad __gcov0.main
.LC1:
.string "/home/ce/./output.gcda"
.LPBX0:
.long 1094267690
.zero 4
.quad 0
.long 131638524
.zero 4
.quad .LC1
.quad __gcov_merge_add
.quad 0
.quad 0
.quad 0
.quad 0
.quad 0
.quad 0
.quad 0
.quad 0
```

```
.long 1
.zero 4
.quad .LPBX1
.LPBX1:
.quad __gcov_.main
_GLOBAL__sub_I_00100_0_main:
push rbp
mov rbp, rsp
mov edi, OFFSET FLAT:.LPBX0
call __gcov_init
pop rbp
ret
_GLOBAL__sub_D_00100_1_main:
push rbp
mov rbp, rsp
call __gcov_exit
pop rbp
ret
```

Some observations:

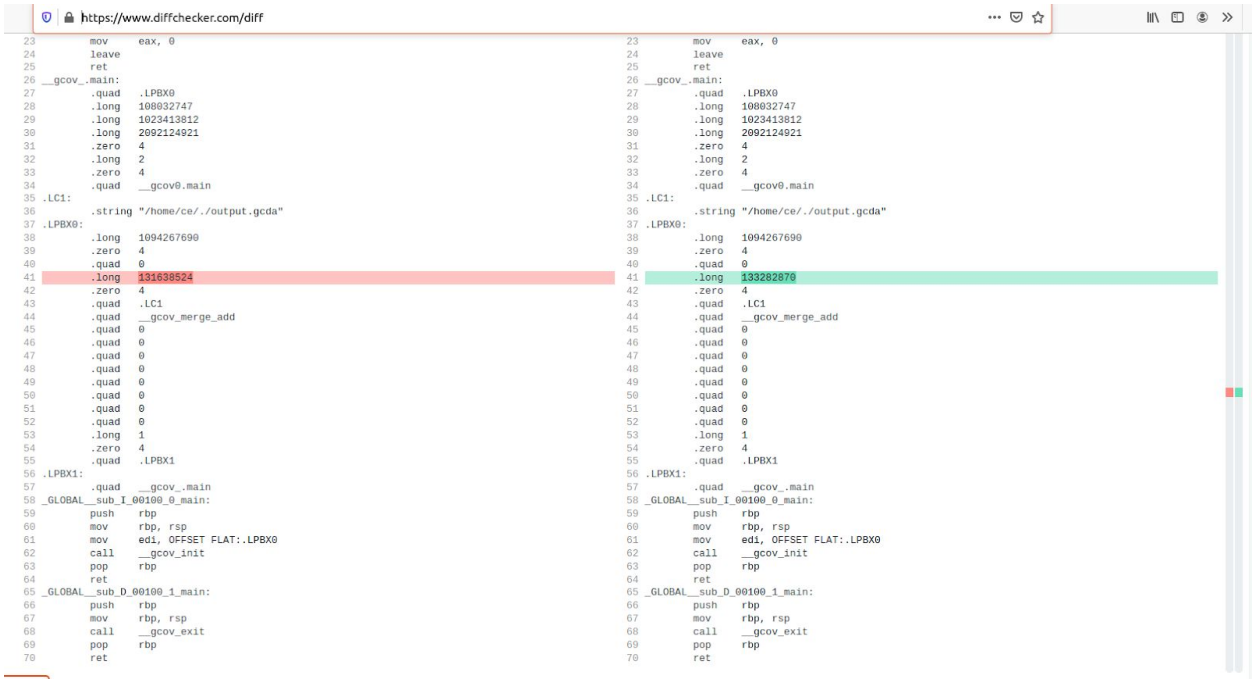
- The compiler has not only generated a separate GCDA file, its also inserted a few statements into the main program body itself. By studying these differences, we can find what is done to generate GCDA files.
- There's a __gcov_main

Adding -ftest-coverage and -branch-probabilities

After this, I decided to check what was happening when the -ftest-coverage and -branch-probabilities options were appended. At first, nothing seemed to be different, the outputs looked identical. However to make sure, I copy-pasted both to https://www.diffchecker.com/ to compare the two text outputs and **there was only one difference**.
Under .LPBX0 on line 41:
For: -c -g -fprofile-arcs toupper.c
.long 131638524

For: -c -g -fprofile-arcs -ftest-coverage
.long 133282870

For: -c -g -fprofile-arcs -fbranch-probabilities
.long 133762972
This was the only difference.



Searching .LPBX0 brought up a couple of old posts regarding ‘gcov weirdness’ such as:

https://gcc.gnu.org/legacy-ml/gcc/2005-07/msg01261.html

https://gcc.gnu.org/legacy-ml/gcc/2005-07/msg01257.html

Hi everyone,

I am getting weird warning messages from my assembler when
gcov is being used. I have tracked what I think is the
problem down but I don't really know how to fix it. The
bit of assembler that causes the warning is:

```
    .type .LPBX0, @object
    .size .LPBX0, 52
.LPBX0:
    ... whole bunch of initialization stuff
```

The assembler warns that the .type/.size directive is useless.
This is causing many testsuite failures. The assembler is
right. Why is a local lable being declared as if it was a
global symbol? .LPBX0 comes from ASM_GENERATE_INTERNAL_LABEL.

The problem seems to start in coverage.c:create_coverage().
Fairly close to the top of that is a call to assemble_variable().
assemble_variable(), ends up calling ASM_DECLARE_OBJECT_NAME.
This is what is emitting the .type and .size directives.

Is there some way that the tree can be marked as an internal
construct, and if that is set, not have the bit of code in
assemble_variable() call ASM_DECLARE_OBJECT_NAME? Or perhaps
I can train A_D_O_N to not emit anything if its a local?

I see that the tree in question is declared static. Would it
be the right thing to have something like this at the end of
assemble_variable:

```
   if (!TREE_STATIC (decl)) {
#ifdef ASM_DECLARE_OBJECT_NAME
     last_assemble_variable_decl = decl;
     ASM_DECLARE_OBJECT_NAME (asm_out_file, name, decl);
#else
     /* Standard thing is just output label for the object.  */
     ASM_OUTPUT_LABEL (asm_out_file, name);
#endif
   }
```

I am not sure if "if (TREE_STATIC (decl))" is the correct test.
Is there a better test that I can check for an internal lable?

Thanks in advance for any insight.

--

There's an article on the net on gcov internals here:
http://samueldotj.com/blog/gcov-internals-overview/

Some more info - the gcov support functions like gcov_init are in:
/usr/lib/gcc/x86_64-linux-gnu/7/libgcov.a

Seems that the sources for gcov are part of libgcc - the gcc support functions.

GCC IR, Libgcov, and further Analysis

Code for Libgcc: https://github.com/gcc-mirror/gcc/tree/master/libgcc

https://github.com/gcc-mirror/gcc/blob/master/libgcc/libgcov.h

```
246
247     /* Pointer to counters.  */
248     gcov_type *counters;
249   };
250
251   /* Exactly one of these will be active in the process.  */
252   extern struct gcov_master __gcov_master;
253   extern struct gcov_kvp __gcov_kvp_pool[GCOV_PREALLOCATED_KVP];
254   extern unsigned __gcov_kvp_pool_index;
255
256   /* Dump a set of gcov objects.  */
257   extern void __gcov_dump_one (struct gcov_root *) ATTRIBUTE_HIDDEN;
258
259   /* Register a new object file module.  */
260   extern void __gcov_init (struct gcov_info *) ATTRIBUTE_HIDDEN;
261
262   /* GCOV exit function registered via a static destructor.  */
263   extern void __gcov_exit (void) ATTRIBUTE_HIDDEN;
```

Many files such as libgcov driver, interface, merge, profile and gcov.h are relevant.

Preprocessor:

Needed to check if the GCC preprocessor was adding anything to the programs. The preprocessor can be invoked with the -E flag. Running:

gcc -E toupper.c
gcc -E fprofile-arcs -ftest-coverage toupper.c

Did not result in any differences among the file. So I concluded that there's no difference in the preprocessing step. This is not surprising, the article mentioned earlier http://samueldotj.com/blog/gcov-internals-overview/ mentions that the counters are inbuilt, so I wouldn't expect this step to do much anyway.

Disassembly:

Just running: gcc -o toupper.o toupper.c generates the object file which is not readable.

To generate the assembly file:
gcc -o toupper.s -S toupper.c

Then to see the disassembly output:
as -o toupper.o toupper.s //assembler
objdump -d toupper.o

Without -fprofile-arcs, the disassembly is:

toupper.o: file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <main>:

```
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   48 83 ec 10             sub    $0x10,%rsp
   c:   c6 45 ff 61             movb   $0x61,-0x1(%rbp)
  10:   0f be 45 ff             movsbl -0x1(%rbp),%eax
  14:   89 c7                   mov    %eax,%edi
  16:   e8 00 00 00 00                  callq  1b <main+0x1b>
  1b:   89 c2                   mov    %eax,%edx
```

```
 1d:   0f be 45 ff              movsbl -0x1(%rbp),%eax
 21:   89 c6                    mov    %eax,%esi
 23:   48 8d 3d 00 00 00 00     lea    0x0(%rip),%rdi        # 2a <main+0x2a>
 2a:   b8 00 00 00 00                  mov    $0x0,%eax
 2f:   e8 00 00 00 00                  callq  34 <main+0x34>
 34:   b8 00 00 00 00                  mov    $0x0,%eax
 39:   c9                       leaveq
 3a:   c3                       retq
```

With -fprofile-arcs -ftest-coverage , a .gcno file is generated. Which means the gcno file being missing on godbolt is a shortcoming of godbolt, which only shows gcda for some reason.

Commands:

```
gcc -fprofile-arcs -ftest-coverage  -o toupper.s -S toupper.c
as -o toupper.o toupper.s
objdump -d toupper.o
```

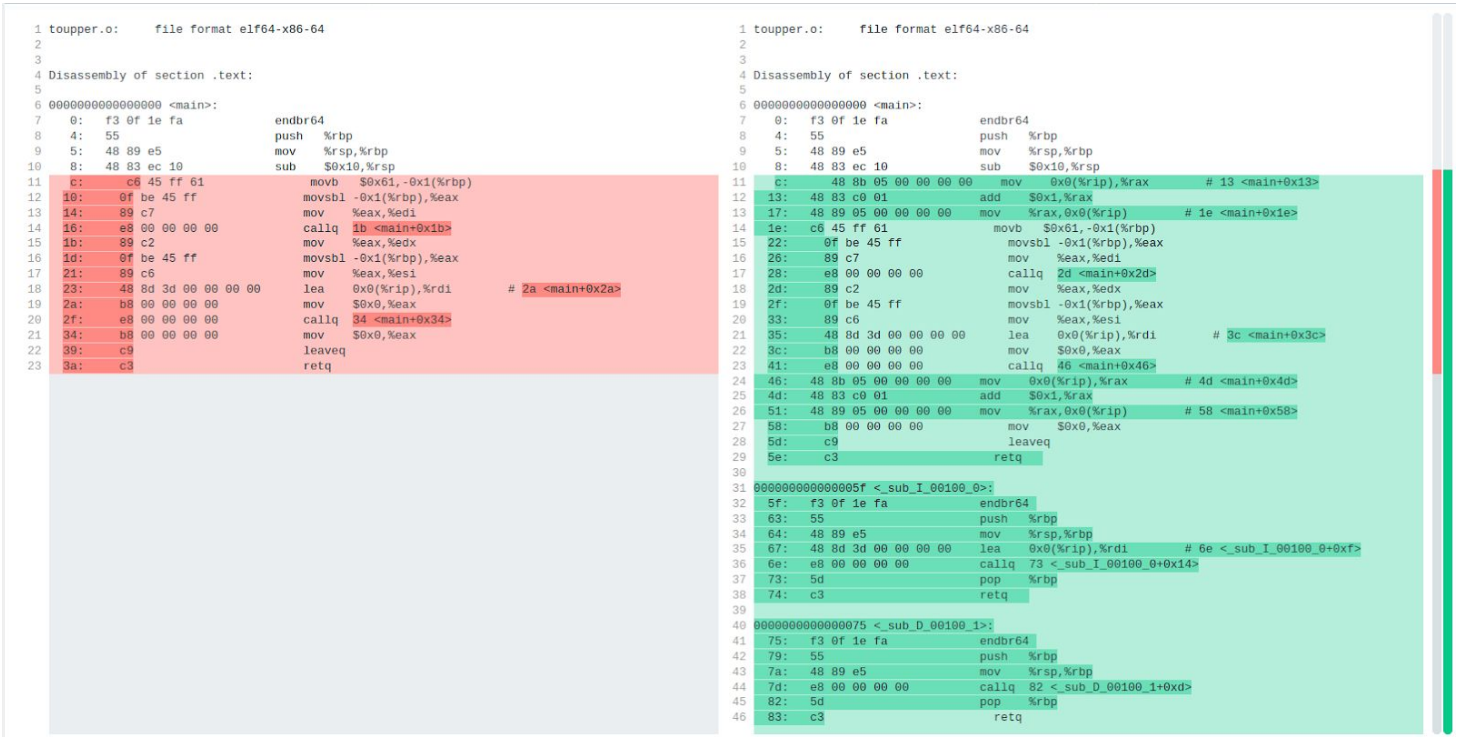
toupper.o: file format elf64-x86-64

Disassembly of section .text:

```
0000000000000000 <main>:
  0:   f3 0f 1e fa               endbr64
  4:   55                        push   %rbp
  5:   48 89 e5                  mov    %rsp,%rbp
  8:   48 83 ec 10               sub    $0x10,%rsp
  c:   48 8b 05 00 00 00 00      mov    0x0(%rip),%rax        # 13 <main+0x13>
 13:   48 83 c0 01               add    $0x1,%rax
 17:   48 89 05 00 00 00 00      mov    %rax,0x0(%rip)        # 1e <main+0x1e>
 1e:   c6 45 ff 61               movb   $0x61,-0x1(%rbp)
 22:   0f be 45 ff               movsbl -0x1(%rbp),%eax
 26:   89 c7                     mov    %eax,%edi
 28:   e8 00 00 00 00                   callq  2d <main+0x2d>
 2d:   89 c2                     mov    %eax,%edx
 2f:   0f be 45 ff               movsbl -0x1(%rbp),%eax
 33:   89 c6                     mov    %eax,%esi
 35:   48 8d 3d 00 00 00 00      lea    0x0(%rip),%rdi        # 3c <main+0x3c>
 3c:   b8 00 00 00 00                   mov    $0x0,%eax
 41:   e8 00 00 00 00                   callq  46 <main+0x46>
 46:   48 8b 05 00 00 00 00      mov    0x0(%rip),%rax        # 4d <main+0x4d>
 4d:   48 83 c0 01               add    $0x1,%rax
```

```
 51:   48 89 05 00 00 00 00     mov  %rax,0x0(%rip)        # 58 <main+0x58>
 58:   b8 00 00 00 00                  mov    $0x0,%eax
 5d:   c9                      leaveq
 5e:   c3                      retq

000000000000005f <_sub_I_00100_0>:
 5f:   f3 0f 1e fa             endbr64
 63:   55                      push   %rbp
 64:   48 89 e5                mov    %rsp,%rbp
 67:   48 8d 3d 00 00 00 00     lea    0x0(%rip),%rdi        # 6e <_sub_I_00100_0+0xf>
 6e:   e8 00 00 00 00                  callq  73 <_sub_I_00100_0+0x14>
 73:   5d                      pop    %rbp
 74:   c3                      retq

0000000000000075 <_sub_D_00100_1>:
 75:   f3 0f 1e fa             endbr64
 79:   55                      push   %rbp
 7a:   48 89 e5                mov    %rsp,%rbp
 7d:   e8 00 00 00 00                  callq  82 <_sub_D_00100_1+0xd>
 82:   5d                      pop    %rbp
 83:   c3                      retq
```

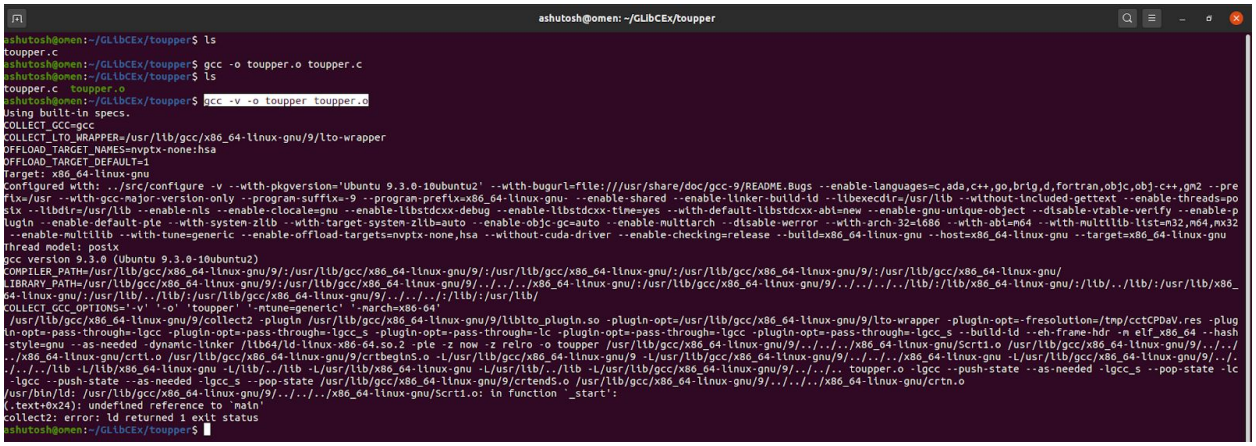
The generated disassembly is twice as long (23 LOC vs 46 LOC). The whole program is altered except for the first few lines. I didn't link the program because I wasn't sure what to link it against.



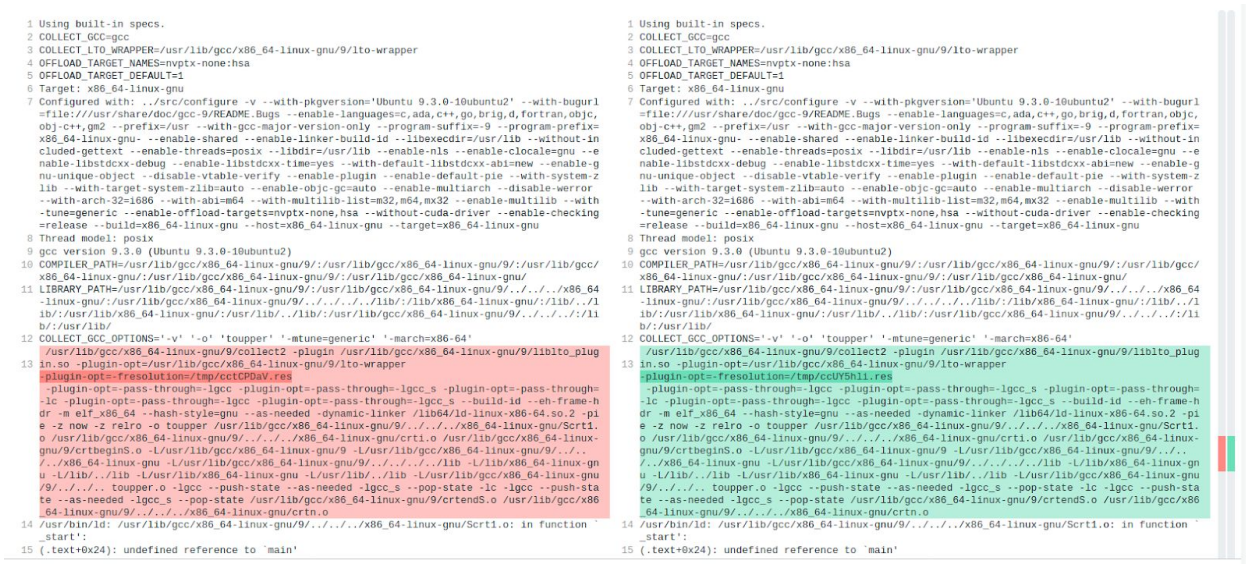
Linker (verbose mode)

Using:
gcc -o toupper.o toupper.c //to generate the object file
And using the -v flag to enable verbose output to see the linking:
gcc -v -o toupper toupper.o

This gives the verbose output:



Trying the same with -fprofile-arcs enabled:



No significant difference, apart from:
-plugin-opt=-fresolution=/tmp/cctCPDaV.res
Changing to:
-plugin-opt=-fresolution=/tmp/ccUY5hli.res

Tree Dump

gcc -fdump-tree-all toupper.c
Produces multiple files

```
ashutosh@omen:~/GLibCEx/toupper$ gcc -fdump-tree-all toupper.c
ashutosh@omen:~/GLibCEx/toupper$ ls
a.out                   toupper.c.008t.lower                toupper.c.018t.fixup_cfg1       toupper.c.043t.profile_estimate  toupper.c.222t.cplxlower0      toupper.c.321t.debug
toupper.c               toupper.c.011t.eh                   toupper.c.019t.ssa              toupper.c.046t.release_ssa       toupper.c.224t.switchlower_O0
toupper.c.004t.original toupper.c.012t.cfg                  toupper.c.023t.fixup_cfg2       toupper.c.047t.local-fnsummary2  toupper.c.231t.optimized
toupper.c.005t.gimple   toupper.c.013t.ompexp               toupper.c.024t.local-fnsummary1 toupper.c.085t.fixup_cfg3        toupper.c.319t.statistics
toupper.c.007t.omplower toupper.c.014t.printf-return-value1 toupper.c.025t.einline          toupper.c.221t.veclower          toupper.c.320t.earlydebug
ashutosh@omen:~/GLibCEx/toupper$ gcc -fprofile-arcs -ftest-coverage -fdump-tree-all toupper.c
ashutosh@omen:~/GLibCEx/toupper$ ls
a.out                   toupper.c.008t.lower                toupper.c.018t.fixup_cfg1       toupper.c.043t.profile_estimate  toupper.c.222t.cplxlower0      toupper.c.321t.debug
toupper.c               toupper.c.011t.eh                   toupper.c.019t.ssa              toupper.c.046t.release_ssa       toupper.c.224t.switchlower_O0  toupper.gcno
toupper.c.004t.original toupper.c.012t.cfg                  toupper.c.023t.fixup_cfg2       toupper.c.047t.local-fnsummary2  toupper.c.231t.optimized
toupper.c.005t.gimple   toupper.c.013t.ompexp               toupper.c.024t.local-fnsummary1 toupper.c.085t.fixup_cfg3        toupper.c.319t.statistics
toupper.c.007t.omplower toupper.c.014t.printf-return-value1 toupper.c.025t.einline          toupper.c.221t.veclower          toupper.c.320t.earlydebug
ashutosh@omen:~/GLibCEx/toupper$
```

A **Control Flow Graph (CFG)** is the graphical representation of control flow or computation during the execution of programs or applications. Control flow graphs are mostly used in static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit.

We can view the CFG with -fdump-tree-cfg flag and this gives a bunch of information about the program:

First:
gcc -fdump-tree-cfg toupper.c
Generates:

```
 1
 2 ;; Function main (main, funcdef_no=0, decl_uid=2425, cgraph_uid=1, symbol_order=0)
 3
 4 ;; 1 loops found
 5 ;;
 6 ;; Loop 0
 7 ;;  header 0, latch 1
 8 ;;  depth 0, outer -1
 9 ;;  nodes: 0 1 2 3
10 ;; 2 succs { 3 }
11 ;; 3 succs { 1 }
12 main ()
13 {
14   char c;
15   int D.2429;
16
17   <bb 2> :
18   c = 97;
19   _1 = (int) c;
20   _2 = toupper (_1);
21   _3 = (int) c;
22   printf ("%c -> %c", _3, _2);
23   D.2429 = 0;
24
25   <bb 3> :
26 <L0>:
27   return D.2429;
28
29 }
30
31
```

Then: gcc -fprofile-arcs -ftest-coverage -fdump-tree-cfg toupper.c
The main part of the program (upto line number 29) is the same but then the following is appended:

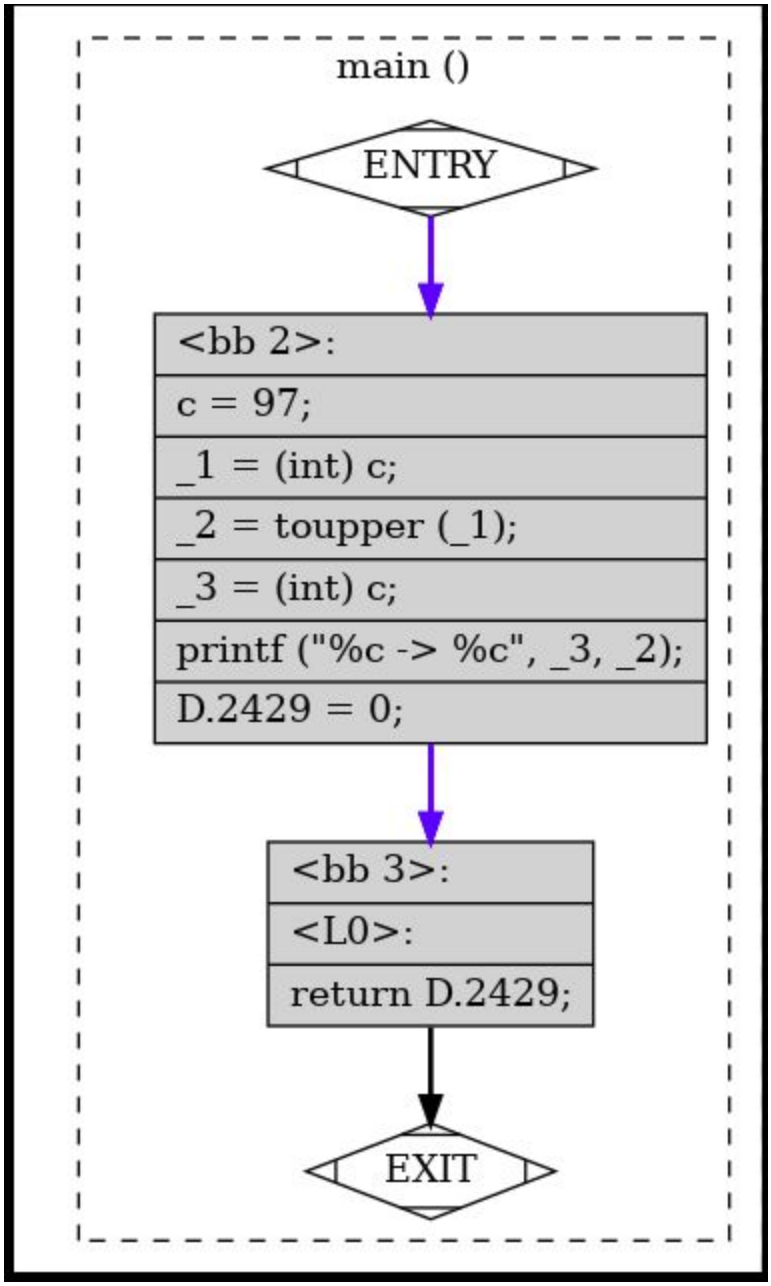
```
}


;; Function _sub_I_00100_0 (_sub_I_00100_0, funcdef_no=1, decl_uid=2463, cgraph_uid=4, symbol_order=5)

;; 1 loops found
;;
;; Loop 0
;;  header 0, latch 1
;;  depth 0, outer -1
;;  nodes: 0 1 2
;; 2 succs { 1 }
_sub_I_00100_0 ()
{
  <bb 2> :
  __gcov_init (&*.LPBX0);
  return;

}


;; Function _sub_D_00100_1 (_sub_D_00100_1, funcdef_no=2, decl_uid=2466, cgraph_uid=5, symbol_order=6)

;; 1 loops found
;;
;; Loop 0
;;  header 0, latch 1
;;  depth 0, outer -1
;;  nodes: 0 1 2
;; 2 succs { 1 }
_sub_D_00100_1 ()
{
  <bb 2> :
  __gcov_exit ();
  return;
```

Graphical Output:
To better understand what is happening, we enable graphical output, we can generate .dot files and output them:
gcc -fdump-tree-all-graph toupper.c -o toupper

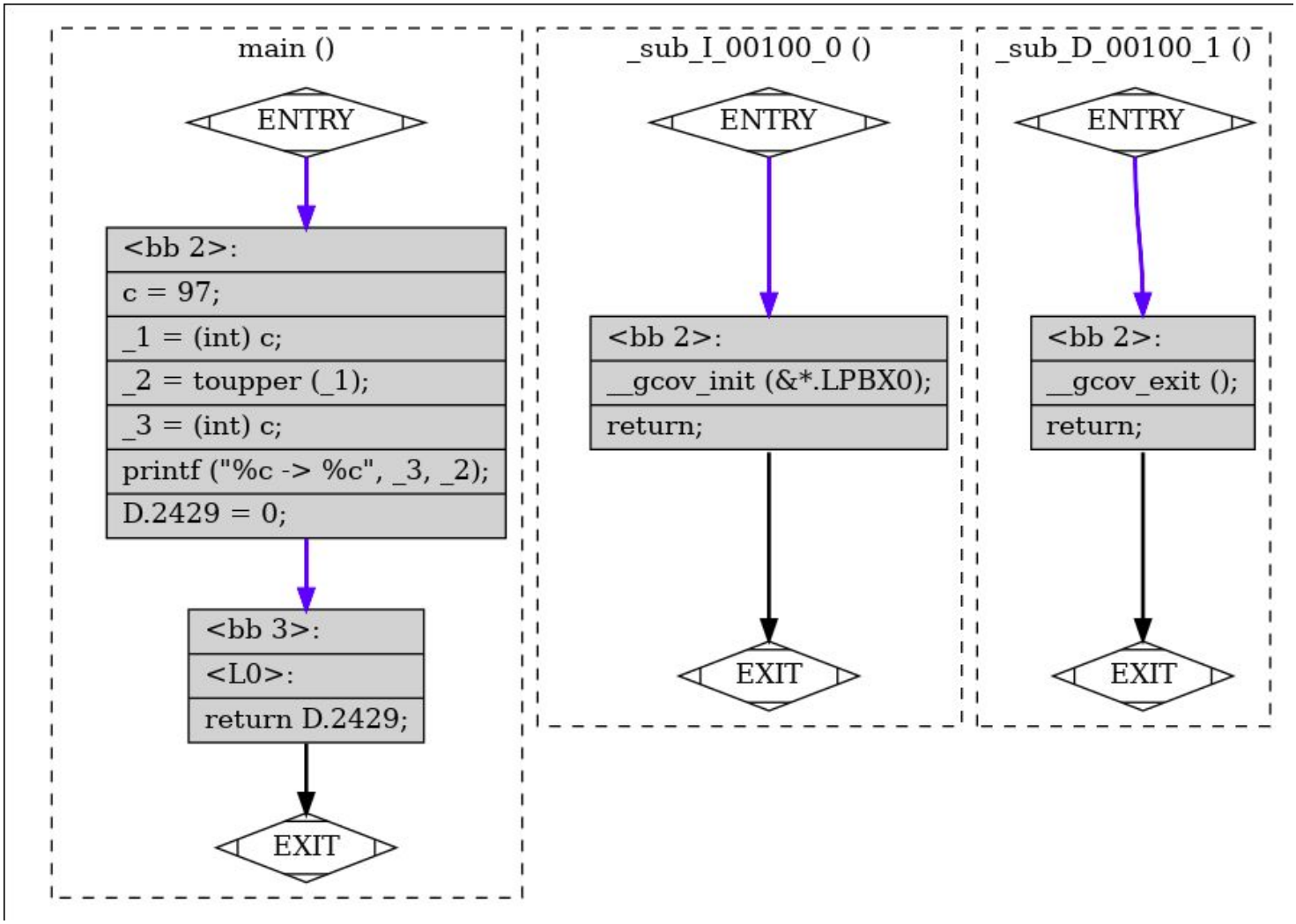
```
ashutosh@omen:~/GLibCEx/toupper$ gcc -fdump-tree-all-graph toupper.c -o toupper
ashutosh@omen:~/GLibCEx/toupper$ ls
a.out
toupper
toupper.c
toupper.c.004t.original
toupper.c.005t.gimple
toupper.c.007t.omplower
toupper.c.008t.lower
toupper.c.011t.eh
toupper.c.012t.cfg
toupper.c.012t.cfg.dot
toupper.c.013t.ompexp
toupper.c.013t.ompexp.dot
toupper.c.014t.printf-return-value1
toupper.c.014t.printf-return-value1.dot
toupper.c.018t.fixup_cfg1
toupper.c.018t.fixup_cfg1.dot
toupper.c.019t.ssa
toupper.c.019t.ssa.dot
toupper.c.023t.fixup_cfg2
toupper.c.023t.fixup_cfg2.dot
toupper.c.024t.local-fnsummary1
toupper.c.024t.local-fnsummary1.dot
toupper.c.025t.einline
toupper.c.025t.einline.dot
toupper.c.043t.profile_estimate
toupper.c.046t.release_ssa
toupper.c.046t.release_ssa.dot
toupper.c.047t.local-fnsummary2
toupper.c.047t.local-fnsummary2.dot
toupper.c.085t.fixup_cfg3
toupper.c.085t.fixup_cfg3.dot
toupper.c.221t.veclower
toupper.c.221t.veclower.dot
toupper.c.222t.cplxlower0
toupper.c.222t.cplxlower0.dot
toupper.c.224t.switchlower_O0
toupper.c.224t.switchlower_O0.dot
toupper.c.231t.optimized
toupper.c.231t.optimized.dot
toupper.c.319t.statistics
toupper.c.320t.earlydebug
toupper.c.321t.debug
toupper.gcno
ashutosh@omen:~/GLibCEx/toupper$ dot
```

After installing graphviz:
sudo apt install graphviz
We run,
dot -Tpng toupper.c.012t.cfg.dot -o cfg1.png

To get the control flow graph for the toupper.c program:

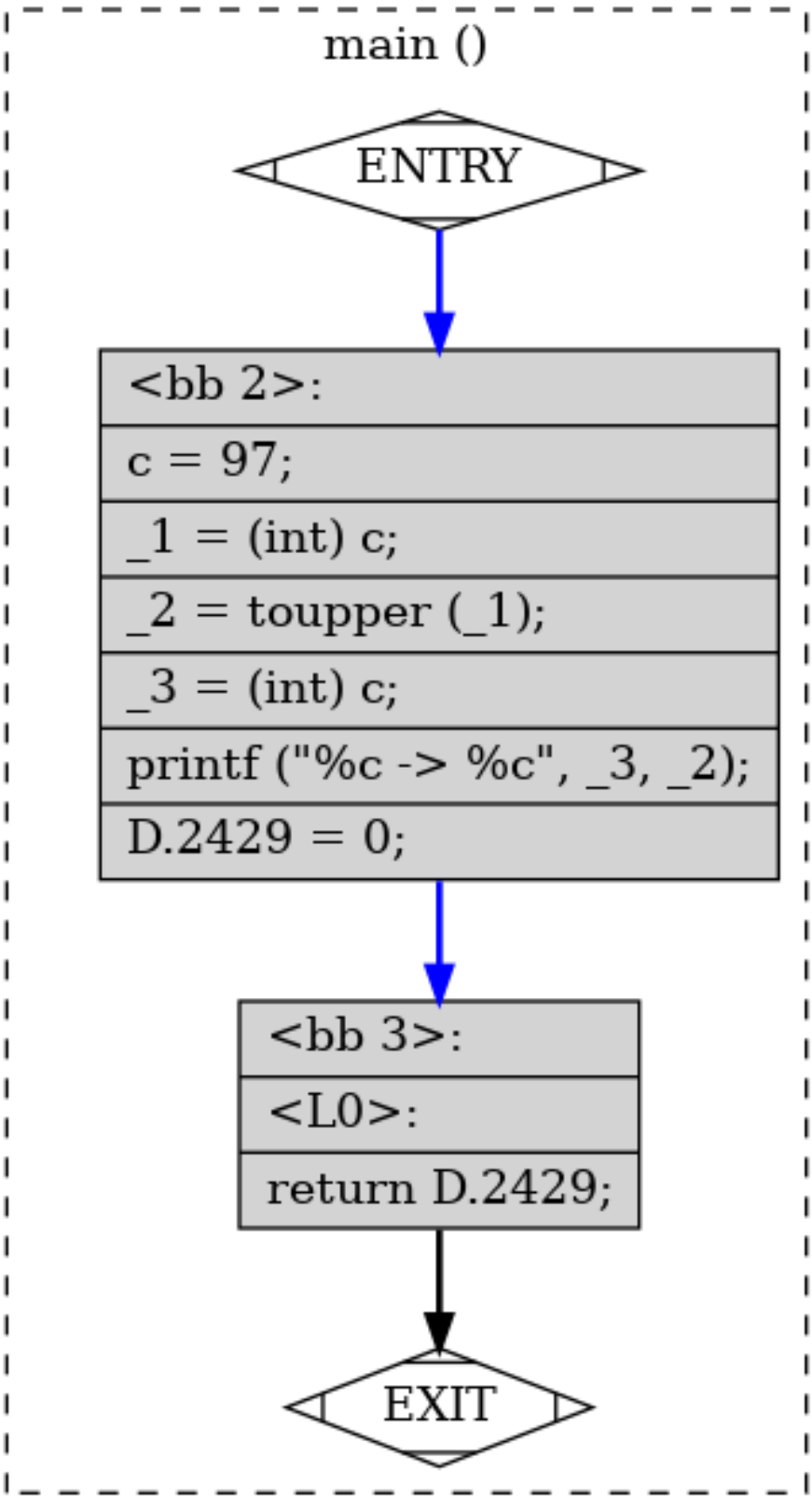


Running:
gcc -fprofile-arcs -ftest-coverage -fdump-tree-all-graph toupper.c -o toupper
Followed by:
dot -Tpng toupper.c.012t.cfg.dot -o cfg1.png
Generates a different CFG:



The other outputs are not very different, the optimized output contains substitutions for variables (to make sure that each variable is labelled properly) but other than that there are no significant differences.

Eg:
dot -Tpng toupper.c.231t.optimized.dot -o cfg1.png

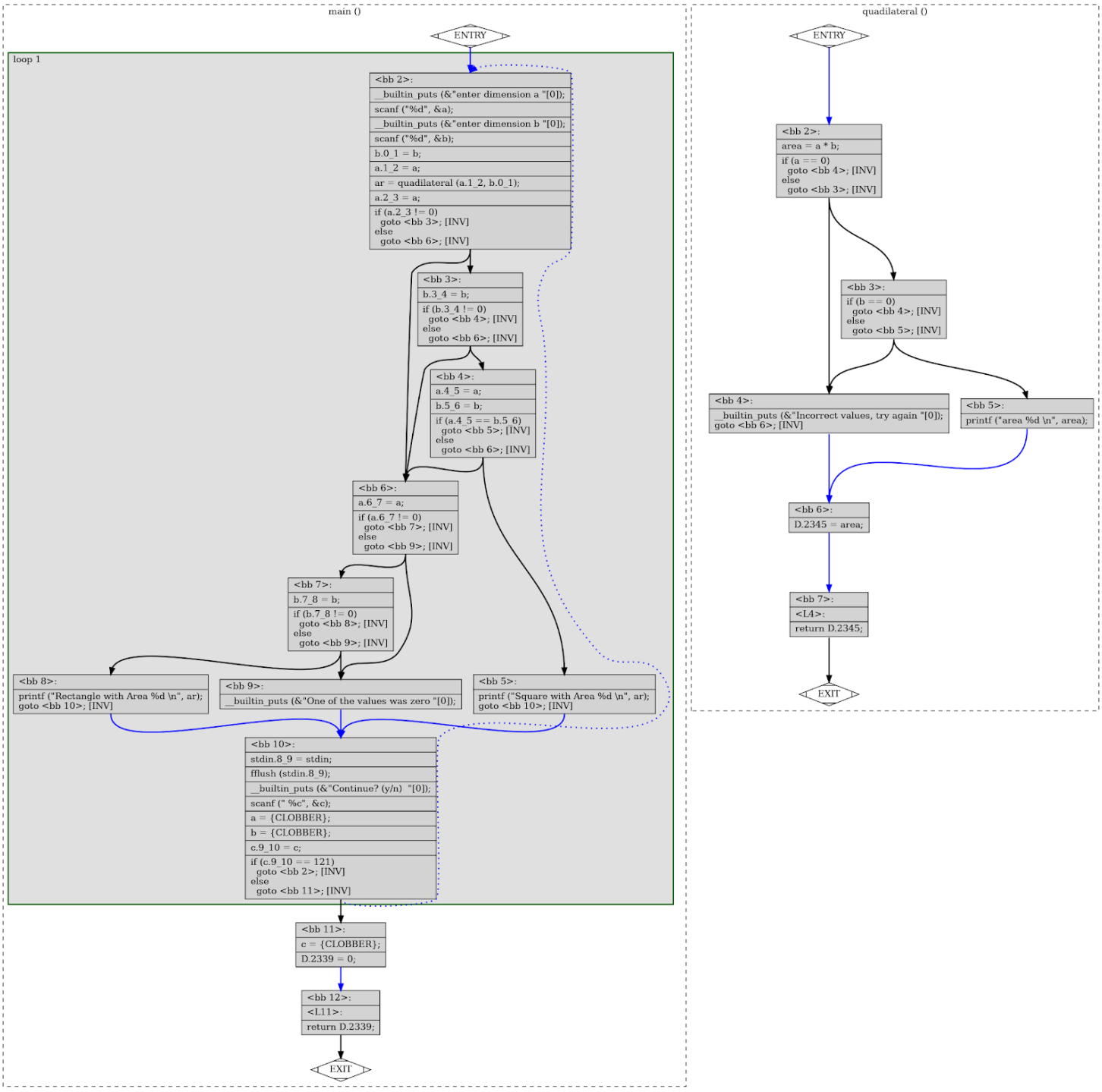

main ()
ENTRY
<bb 2>:
c = 97;
_1 = (int) c;
_2 = toupper (_1);
_3 = (int) c;
printf ("%c -> %c", _3, _2);
D.2429 = 0;
<bb 3>:
<L0>:
return D.2429;
EXIT

Running More complex examples:

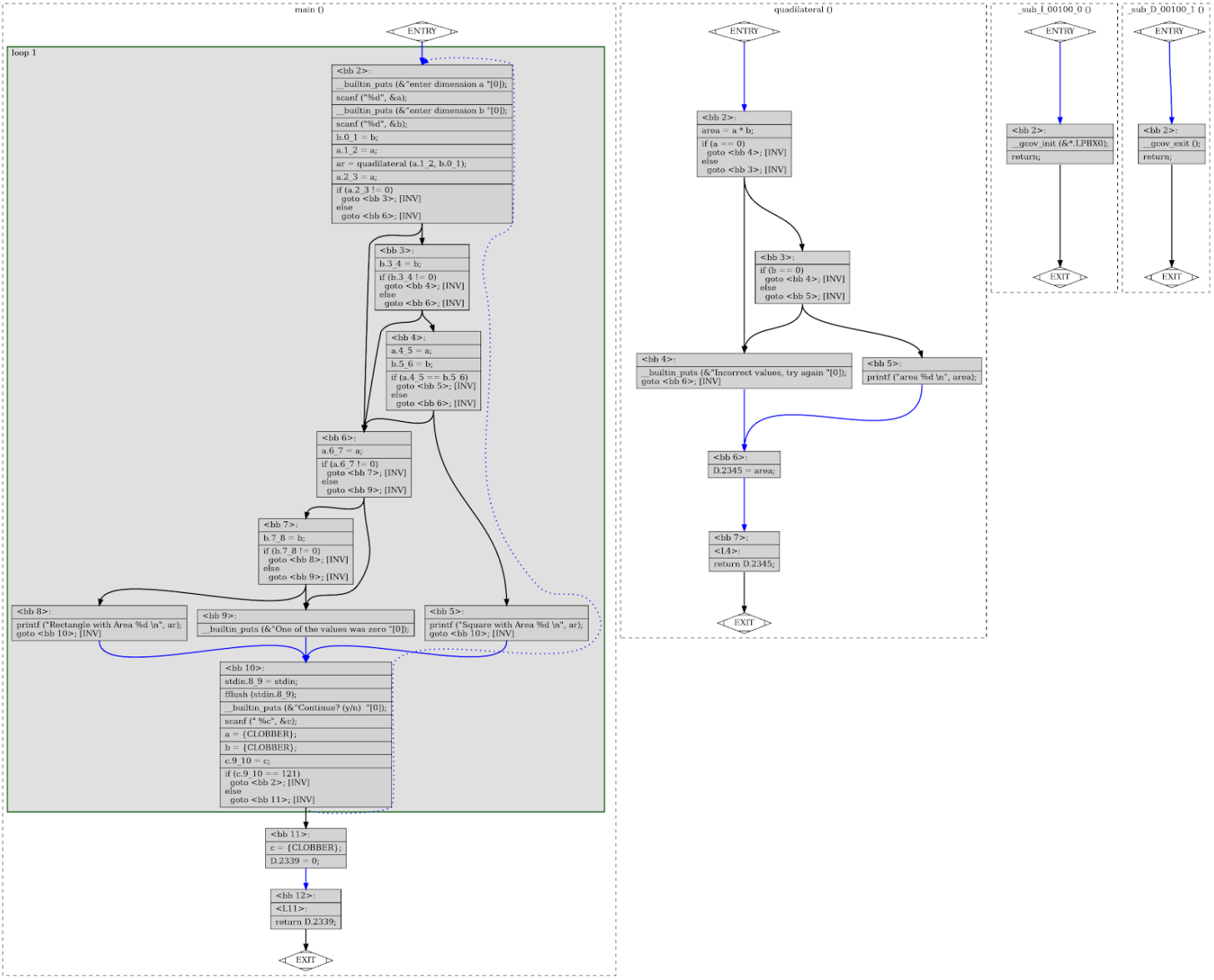
Using the more complicated program with branching:

```
/* Program takes the input dimensions A and B, and computes the area.
Incorrect values of A or B that do not result in a square or rectangle are rejected.
Then it is classified as square or rectangle */
#include <stdio.h>
int quadilateral(int a, int b);
int main()
{
char c;
do {
int a;
int b;
printf("enter dimension a \n");
scanf("%d", &a);
printf("enter dimension b \n");
scanf("%d", &b);
int ar = quadilateral(a,b);
If ( !(a== 0 || b== 0 )&& ( a == b ))
printf("Square with Area %d \n", ar);
else if ( ! ( a == 0 || b == 0 ))
printf("Rectangle with Area %d \n", ar);
else
printf("One of the values was zero \n");
fflush(stdin);
printf("Continue? (y/n) \n");
scanf(" %c", &c);
}
while(c == 'y');
}
int quadilateral(int a,int b)
{
int area = a*b;
if(a == 0 || b == 0 )
printf("Incorrect values, try again \n");
else
printf("area %d \n", area);
return area;
}
```

Generating CFG:



With -fprofile-arcs and -ftest-coverage



Assembly Output:

An extra section labelled _gcov_quadilateral is created. With its own LPBX0.

```
__gcov_.quadilateral(int, int):
.quad .LPBX0
.long 463939559
.long -1662633249
.long 1677313757
.zero 4
.long 5
.zero 4
.quad __gcov0.quadilateral(int, int)
```

Viewing symbols:

.i file is the preprocessed file.

```
$ cpp example1.c -o example1.i
$ gcc -fprofile-arcs -ftest-coverage -S example1.i
$ as -o example1.o example1.s
$ gcc -o example1 example1.o -lmygcov
```

Output:

```
/usr/bin/ld: example1.o: in function `_sub_I_00100_0':
example1.c:(.text+0x332): undefined reference to `__gcov_init'
/usr/bin/ld: example1.o: in function `_sub_D_00100_1':
example1.c:(.text+0x341): undefined reference to `__gcov_exit'
/usr/bin/ld: example1.o:(.data.rel+0x20): undefined reference to `__gcov_merge_add'
collect2: error: ld returned 1 exit status
```

```
$ nm example1.o
```

```
                 U fflush
0000000000000040 b __gcov0.main
0000000000000000 b __gcov0.quadilateral
                 U __gcov_exit
                 U __gcov_init
0000000000000040 d __gcov_.main
                 U __gcov_merge_add
0000000000000000 d __gcov_.quadilateral
                 U _GLOBAL_OFFSET_TABLE_
                 U __isoc99_scanf
0000000000000000 T main
                 U printf
                 U puts
0000000000000277 T quadilateral
                 U __stack_chk_fail
                 U stdin
0000000000000338 t _sub_D_00100_1
0000000000000322 t _sub_I_00100_0
```

```
ashutosh@ashu:~/Desktop/shmEx$ cpp example1.c -o example1.i
ashutosh@ashu:~/Desktop/shmEx$ gcc -fprofile-arcs -ftest-coverage-S example1.i
gcc: error: unrecognized command line option '-ftest-coverage-S'; did you mean '-ftest-coverage'?
ashutosh@ashu:~/Desktop/shmEx$ gcc -fprofile-arcs -ftest-coverage -S example1.i
ashutosh@ashu:~/Desktop/shmEx$ as -o example1.o example1.s
ashutosh@ashu:~/Desktop/shmEx$ gcc -o example1 example1.o
/usr/bin/ld: example1.o: in function `_sub_I_00100_0':
example1.c:(.text+0x332): undefined reference to `__gcov_init'
/usr/bin/ld: example1.o: in function `_sub_D_00100_1':
example1.c:(.text+0x341): undefined reference to `__gcov_exit'
/usr/bin/ld: example1.o:(.data.rel+0x20): undefined reference to `__gcov_merge_add'
collect2: error: ld returned 1 exit status
ashutosh@ashu:~/Desktop/shmEx$ nm example1.o
                 U fflush
0000000000000040 b __gcov0.main
0000000000000000 b __gcov0.quadilateral
                 U __gcov_exit
                 U __gcov_init
0000000000000040 d __gcov_.main
                 U __gcov_merge_add
0000000000000000 d __gcov_.quadilateral
                 U _GLOBAL_OFFSET_TABLE_
                 U __isoc99_scanf
0000000000000000 T main
                 U printf
                 U puts
0000000000000277 T quadilateral
                 U __stack_chk_fail
                 U stdin
0000000000000338 t _sub_D_00100_1
0000000000000322 t _sub_I_00100_0
```

.LC2, .LC3, .LPBX0 main other codes are inserted into the program.

Switching back to our shorter program for ease of readability:

nm toupper.o
0000000000000000 b __gcov0.main
                 U __gcov_exit
                 U __gcov_init
0000000000000000 d __gcov_.main
                 U __gcov_merge_add
                 U _GLOBAL_OFFSET_TABLE_
0000000000000000 T main
                 U printf
0000000000000075 t _sub_D_00100_1
000000000000005f t _sub_I_00100_0
                 U toupper
We see the same symbols, so size of program doesn't matter

https://titanwolf.org/Network/Articles/Article?AID=deb3199a-b3c3-4ab0-b6d8-9d79ef8f87a3

https://docs.huihoo.com/doxygen/linux/kernel/3.7/gcov_8h_source.html

```
81 struct gcov_info {
82  unsigned int version;
83  struct gcov_info *next;
84  unsigned int stamp;
85  const char *filename;
86  unsigned int n_functions;
87  const struct gcov_fn_info *functions;
88  unsigned int ctr_mask;
89  struct gcov_ctr_info counts[0];
90 };
91
```

https://docs.huihoo.com/doxygen/linux/kernel/3.7/gcov_8h_source.html

https://github.com/gcc-mirror/gcc/blob/master/libgcc/gcov.h

Method of passing gcov information without using glibc

https://www.man7.org/linux/man-pages/man3/shm_open.3.html

We move to the next phase of development, which is writing a support library for GCoV that will not use GLibC. One way to do this is to use shared memory. Here, we build the examples:

Step 1: Setting up the files:

The example consists of 3 files, a header file known as pshm_ucase.h , and a pshm_ucase_bounce.c file, along with a pshm_ucase_send.c file. First we create a directory:

```
mkdir shmEx
cd shmEx
```

and create 3 files in it.

Step 2: Compiling:

The pshm_ucase_bounce.c file is supposed to be executed first, so we run:

$gcc pshm_ucase_bounce.c

This returns the following errors:

```
ashutosh@ashu:~/Desktop/shmEx$ gcc pshm_ucase_bounce.c
/usr/bin/ld: /tmp/ccvA3yf2.o: in function `main':
pshm_ucase_bounce.c:(.text+0x63): undefined reference to `shm_open'
/usr/bin/ld: pshm_ucase_bounce.c:(.text+0x108): undefined reference to `sem_init'
/usr/bin/ld: pshm_ucase_bounce.c:(.text+0x13d): undefined reference to `sem_init'
/usr/bin/ld: pshm_ucase_bounce.c:(.text+0x164): undefined reference to `sem_wait'
/usr/bin/ld: pshm_ucase_bounce.c:(.text+0x1d6): undefined reference to `sem_post'
/usr/bin/ld: pshm_ucase_bounce.c:(.text+0x1fd): undefined reference to `shm_unlink'
collect2: error: ld returned 1 exit status
```

After a bit of searching:

https://stackoverflow.com/questions/9923495/undefined-reference-shm-open-already-add-lrt-flag-here

https://stackoverflow.com/questions/26464583/undefined-reference-to-symbol-sem-postglibc-2-4
https://stackoverflow.com/questions/23556042/undefined-reference-issues-using-semaphores

It turns out we need to add a few flags while compiling, the **-pthread flag** and the **-lrt flag**. The pthread flag is used to solve the issue of undefined references to semaphores, while the lrt flag is used to include library rt which is for shm_open and shm_unlink. The -lrt flag is **position sensitive** and must be added to the rightmost end of the compilation command, adding it anywhere else will not work.

So first I run:

$gcc -pthread pshm_ucase_bounce.c -lrt

And the program compiles without any errors, however when I run
$./a.out
I get:
Usage: ./a.out /shm-path

Similarly,
$gcc -pthread pshm_ucase_send.c -lrt
$./a.out
Usage: ./a.out /shm-path string

```
ashutosh@ashu:~/Desktop/shmEx$ gcc -pthread pshm_ucase_bounce.c -lrt
ashutosh@ashu:~/Desktop/shmEx$ ls
a.out  pshm_ucase_bounce.c  pshm_ucase.h  pshm_ucase_send  pshm_ucase_send.c
ashutosh@ashu:~/Desktop/shmEx$ ./a.out
Usage: ./a.out /shm-path
ashutosh@ashu:~/Desktop/shmEx$ gcc -pthread pshm_ucase_send.c -lrt
ashutosh@ashu:~/Desktop/shmEx$ ./a.out
Usage: ./a.out /shm-path string
ashutosh@ashu:~/Desktop/shmEx$
```

However, given that the example on the manpage mentions pshm_ucase_bounce and pshm_ucase_send, when I run:

$ gcc -pthread -o pshm_ucase_bounce pshm_ucase_bounce.c -lrt
$ gcc -pthread -o pshm_ucase_send pshm_ucase_send.c -lrt
$ ./pshm_ucase_bounce /shmEx &
$ ./pshm_ucase_send /shmEx hello

I get the output:

[2] 7769

HELLO
[2]- Done
Which is the expected output

```
ashutosh@ashu:~/Desktop/shmEx$ gcc -pthread -o pshm_ucase_bounce pshm_ucase_bounce.c -lrt
ashutosh@ashu:~/Desktop/shmEx$ gcc -pthread -o pshm_ucase_send pshm_ucase_send.c -lrt
ashutosh@ashu:~/Desktop/shmEx$ ls
a.out  pshm_ucase_bounce  pshm_ucase_bounce.c  pshm_ucase.h  pshm_ucase_send  pshm_ucase_send.c
ashutosh@ashu:~/Desktop/shmEx$ ./pshm_ucase_bounce /shmEx &
[2] 7769
ashutosh@ashu:~/Desktop/shmEx$ ./pshm_ucase_send /shmEx hello
HELLO
[2]-  Done                    ./pshm_ucase_bounce /shmEx
```

Write our own empty implementation of gcov.h - start with __gcov_init and __gcov_exit.

Create a library libmygcov.a that includes the implementation of these empty functions.

Link and executable (from example1.c above) with -lmygov.

Ideas to read/write coverage data without glibc:

1. Use shared memory to another application
2. Write a Linux device driver that will read/write to some filesystem

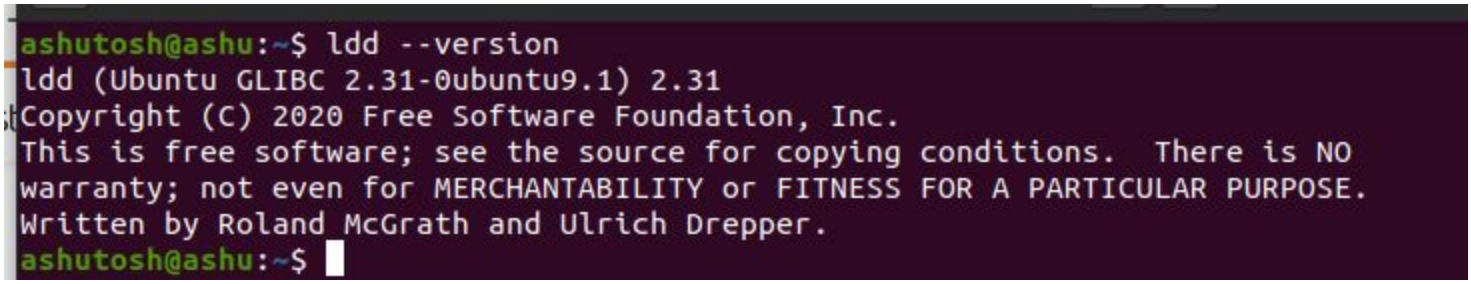
Building GlibC from source and Linking it to GCC

From: https://sourceware.org/glibc/wiki/Testing/Builds?action=recall&rev=21#Compile_against_glibc_in_an_installed_location

“Please note that the installed glibc is an incomplete C runtime. In order to complete the C runtime you may need to copy in additional headers that match the compiler you are using since the use of --sysroot will restrict their lookup to the sysroot.”

T$ ldd --version

ldd (Ubuntu GLIBC 2.31-0ubuntu9.1) 2.31



From: https://docs.oracle.com/cd/E88353_01/html/E37853/crti.o-7.html

“crt1.o provides the _start symbol that the runtime linker, ld.so.1, jumps to in order to pass control to the executable, and is responsible for providing ABI mandated symbols and other process initialization, for calling main(), and ultimately, exit(). crti.o and crtn.o provide prologue and epilogue .init and .fini sections to encapsulate ELF init and fini code.

crt1.o is only used when building executables. crti.o and crtn.o are used by executables and shared objects.

These CRT objects are compatible with position independent (PIC), and position dependent (non-PIC) code, including both normal and position independent executables (PIE).”

Crt0.o, crt1.o, crti.o, crtn.o, crtbegin.o, crtendS.o

These C runtime files (CRT) are linked into an execution file. The role of these files is to start, initialize, construct, destruct and end and usually they are automatically linked by the compiler.

For instance, main() is called through these files. Since we are not doing standard linking (compilation option -nostdlib) we have to specify these object files seperately. If they're not specified, the linker will prompt that start symbol can't be found and the compilation fails. The order in which these runtime files are called is also important because the GNU linker is a single process linker.

GlibC has several run-time libraries (which might be a source of conflict) /usr/lib/crt1.o , /usr/lib/crti.o , /usr/lb/crtn.o etc.

So in our Glibc build, I use the crt1.o, crti.o and crtn.o provided by the host (preinstalled) glibc.

https://sourceware.org/glibc/wiki/Testing/Builds?action=recall&rev=21#Compile_against_glibc_in_an_installed_location

Relevant link:
https://titanwolf.org/Network/Articles/Article?AID=3c6d039f-47b5-498b-9106-0a9e607924d5#gsc.tab=0

Building GLibC

Commands:

//Use export command to mark environment variables to export to the process.

$ export glibc_install="$(pwd)/glibc/build/install"

// getting the source. The latest version of GLibC is 2.32, while ubuntu 20.04 comes with 2.31.

$ git clone git://sourceware.org/git/glibc.git

$ cd glibc

$ git checkout glibc-2.32

$ mkdir build

$ cd build

$ ../configure --prefix "$glibc_install" //referring to the path we had specified earlier

$ make -j `nproc` //nproc prints the number of processing units available. Using it like this will enable all of them to be used to make glibc, this speeds up the process on my machine.

$ make install -j `nproc`

And Glibc is installed.

I got a sample program with a script to run it from this question on stackoverflow: https://stackoverflow.com/questions/10412684/how-to-compile-my-own-glibc-c-standard-library-from-source-and-use-it

test_glibc.c

```
#define _GNU_SOURCE
#include <assert.h>
#include <gnu/libc-version.h>
#include <stdatomic.h>
#include <stdio.h>
#include <threads.h>

atomic_int acnt;
int cnt;

int f(void* thr_data) {
    for(int n = 0; n < 1000; ++n) {
        ++cnt;
        ++acnt;
    }
    return 0;
}

int main(int argc, char **argv) {
    /* Basic library version check. */
    printf("gnu_get_libc_version() = %s\n", gnu_get_libc_version());

    thrd_t thr[10];
    for(int n = 0; n < 10; ++n)
        thrd_create(&thr[n], f, NULL);
    for(int n = 0; n < 10; ++n)
        thrd_join(thr[n], NULL);
    printf("The atomic counter is %u\n", acnt);
    printf("The non-atomic counter is %u\n", cnt);
}
```

Test_glibc.sh

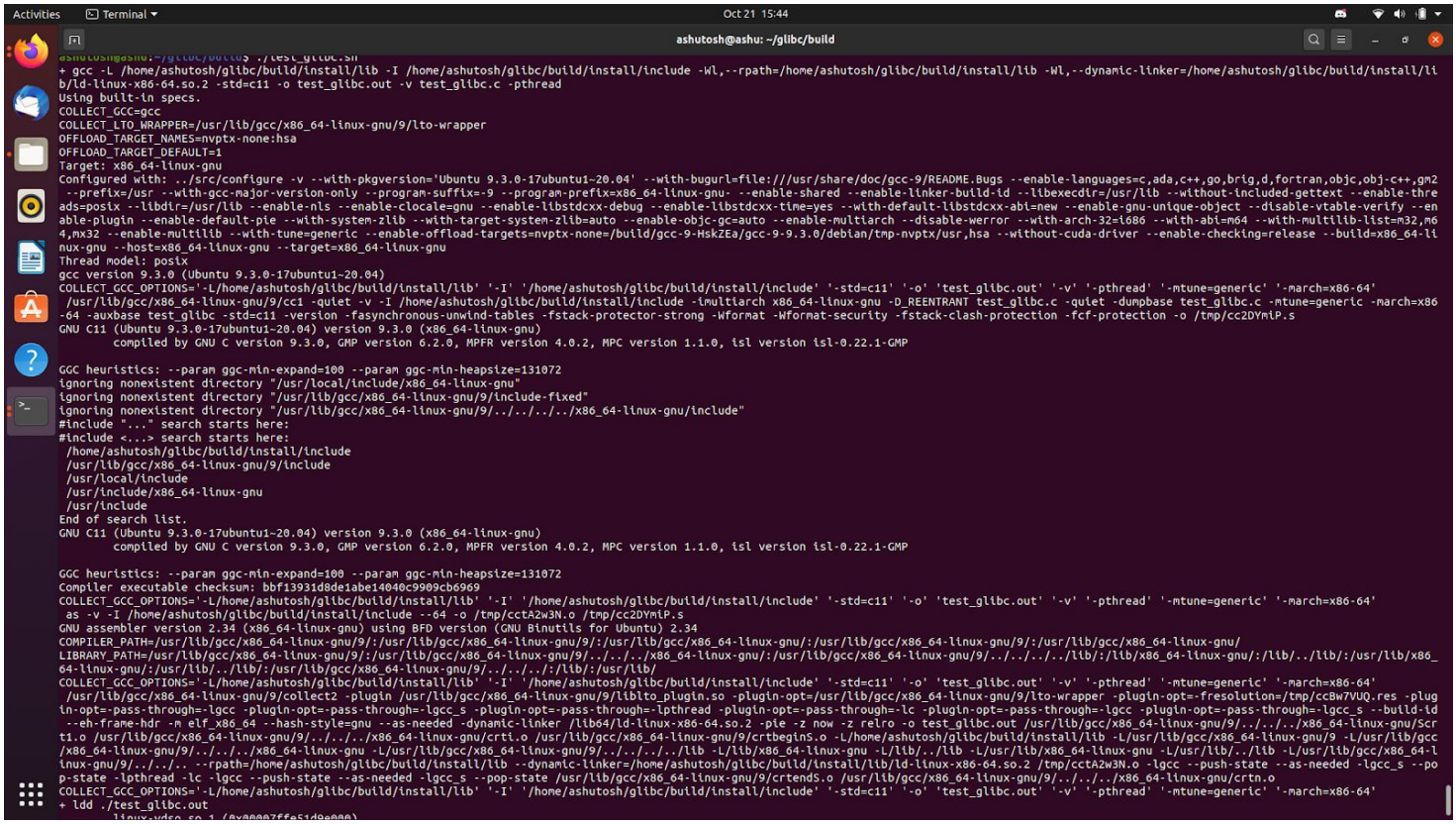
```
#!/usr/bin/env bash

set -eux

gcc \
  -L "${glibc_install}/lib" \
  -I "${glibc_install}/include" \
  -Wl,--rpath="${glibc_install}/lib" \
  -Wl,--dynamic-linker="${glibc_install}/lib/ld-linux-x86-64.so.2" \
  -std=c11 \
  -o test_glibc.out \
  -v \
  test_glibc.c \
  -pthread \
;

ldd ./test_glibc.out

./test_glibc.out
```

Output

```
gnu_get_libc_version() = 2.32

The atomic counter is 10000

The non-atomic counter is 7686
```

Check:

```
gnu_get_libc_version() = 2.32
The atomic counter is 10000
The non-atomic counter is 7686
ashutosh@ashu:~/glibc/build$ ldd --version
ldd (Ubuntu GLIBC 2.31-0ubuntu9.1) 2.31
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```



$ chmod +x test_glibc.sh
$ ./test_glibc.sh
$ ldd test_glibc.out

```
ashutosh@ashu:~/glibc/build$ ldd test_glibc.out
        linux-vdso.so.1 (0x00007ffdfc9fb000)
        libpthread.so.0 => /home/ashutosh/glibc/build/install/lib/libpthread.so.0 (0x00007f296af0f000)
        libc.so.6 => /home/ashutosh/glibc/build/install/lib/libc.so.6 (0x00007f296ad4a000)
        /home/ashutosh/glibc/build/install/lib/ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.2 (0x00007f296af37000)
ashutosh@ashu:~/glibc/build$
```

This proves that our own version of GlibC is being used.

cd glibc/build
make -j `nproc`
make -j `nproc` install
./test_glibc.sh

Using these 4 commands we can build and recompile glibc

Using https://wiki.dlang.org/GDC/Cross_Compiler/crosstool-NG it is possible to build a complete runtime, but I have to test this out.

Next steps: October 21:

1. Document the steps that we did for building glibc with code coverage instrumentation --Done
2. Remove the build_gcov directory and start over again - we had to run configure several times, without -fprofile-arcs and with this flag. Find a way to run configure just once.
3. Fix the configure command so that glibc build can run to the end
    a. Add a -lgcov and after it another link with libc
    b. Do “make install” so that all files are copied to the install directory
4. Write a test program that just calls malloc -Done
5. Find out how to run gcov on this test program so that we can see the file malloc/malloc.c.gcov - the annotated source files (which shows how many times each line and branch were called).

Testing

$ mkdir build_gcov
$ cd build_gcov
$ ../configure --prefix=/home/ashutosh/glibc/build_gcov/install --disable-shared CFLAGS='-fprofile-arcs -ftest-coverage -O2'
$ make
$ cd ..
$ cd build
$ ls
$ cd ..
$ cd build_gcov
$ whereis libc-modules.h
$ cd build_gcov
$ ../configure --prefix=/home/ashutosh/glibc/build_gcov/install --disable-shared CFLAGS= "-fprofile-arcs -ftest-coverage -O2"
$ make

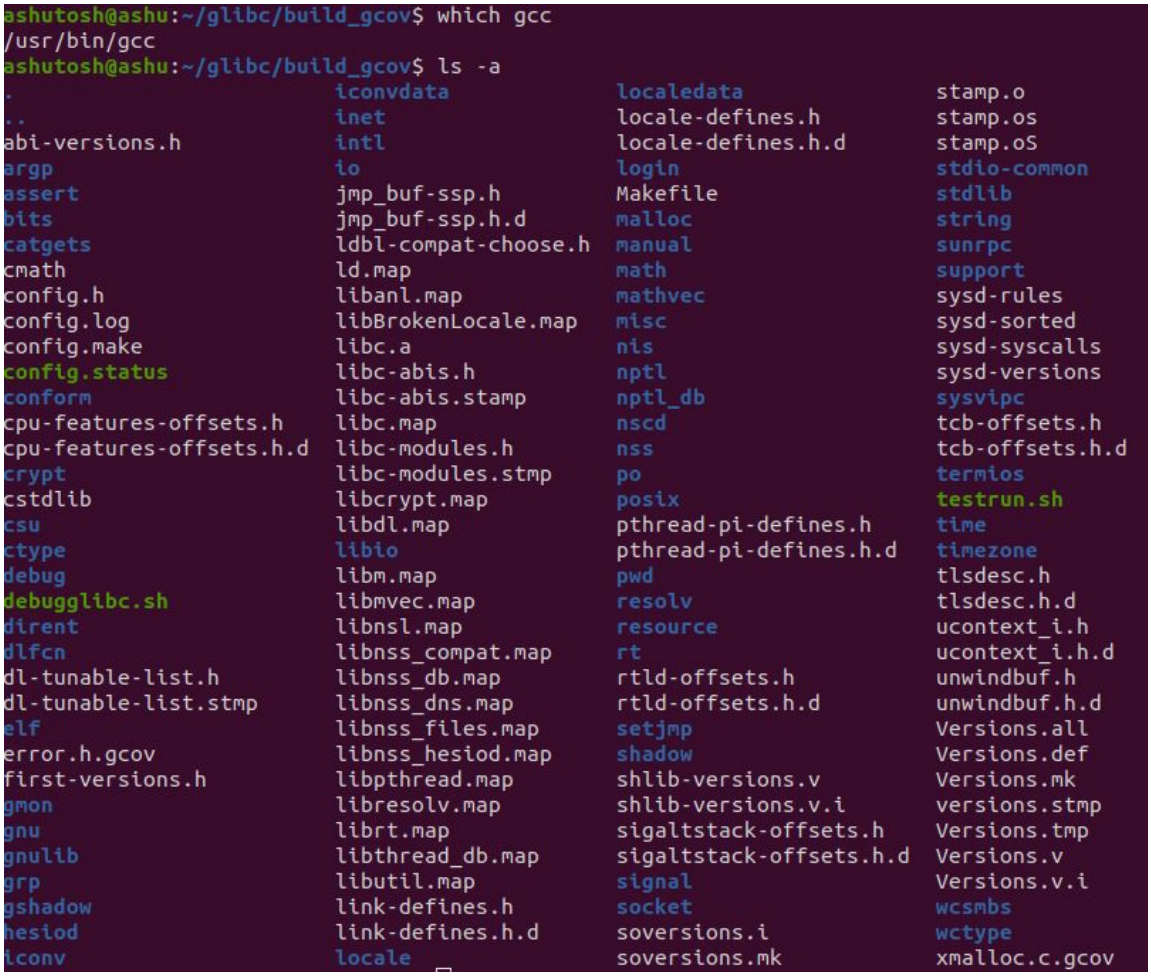
usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(setitimer.o):(.data.rel+0x20): undefined reference to `__gcov_merge_add'
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o): in function `_sub_I_00100_0':
dl-iteratephdr.c:(.text.startup+0xc): undefined reference to `__gcov_init'
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o): in function `_sub_D_00100_1':

dl-iteratephdr.c:(.text.exit+0x5): undefined reference to `__gcov_exit'
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o):(.data.rel+0x20): undefined reference to `__gcov_merge_add'
collect2: error: ld returned 1 exit status
make[2]: *** [../Rules:215: /home/ashutosh/glibc/build_gcov/iconv/iconv_prog] Error 1
make[2]: Leaving directory '/home/ashutosh/glibc/iconv'
make[1]: *** [Makefile:470: iconv/others] Error 2
make[1]: Leaving directory '/home/ashutosh/glibc'
make: *** [Makefile:9: all] Error 2

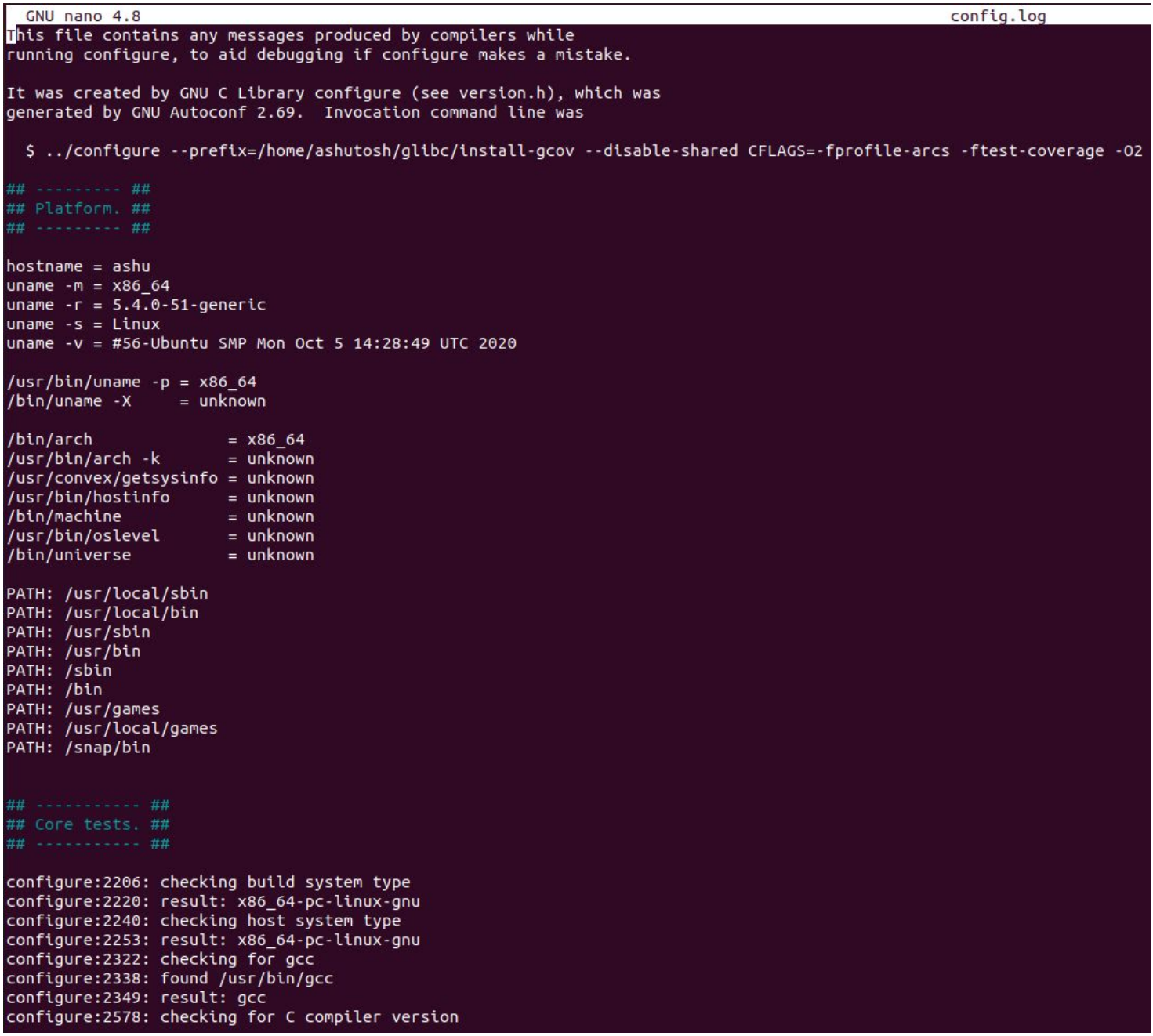
$ which gcc

/usr/bin/gcc

$ ls -a



$ vi config.log



```
  GNU nano 4.8                                                    config.log
This file contains any messages produced by compilers while
running configure, to aid debugging if configure makes a mistake.

It was created by GNU C Library configure (see version.h), which was
generated by GNU Autoconf 2.69.  Invocation command line was

  $ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS=-fprofile-arcs -ftest-coverage -O2

## --------- ##
## Platform. ##
## --------- ##

hostname = ashu
uname -m = x86_64
uname -r = 5.4.0-51-generic
uname -s = Linux
uname -v = #56-Ubuntu SMP Mon Oct 5 14:28:49 UTC 2020

/usr/bin/uname -p = x86_64
/bin/uname -X     = unknown

/bin/arch              = x86_64
/usr/bin/arch -k       = unknown
/usr/convex/getsysinfo = unknown
/usr/bin/hostinfo      = unknown
/bin/machine           = unknown
/usr/bin/oslevel       = unknown
/bin/universe          = unknown

PATH: /usr/local/sbin
PATH: /usr/local/bin
PATH: /usr/sbin
PATH: /usr/bin
PATH: /sbin
PATH: /bin
PATH: /usr/games
PATH: /usr/local/games
PATH: /snap/bin


## ----------- ##
## Core tests. ##
## ----------- ##

configure:2206: checking build system type
configure:2220: result: x86_64-pc-linux-gnu
configure:2240: checking host system type
configure:2253: result: x86_64-pc-linux-gnu
configure:2322: checking for gcc
configure:2338: found /usr/bin/gcc
configure:2349: result: gcc
configure:2578: checking for C compiler version
```

$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS="-fprofile-arcs -ftest-coverage -O2"

$ gcc -- version

```
ashutosh@ashu:~/glibc/build_gcov$ gcc --version
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

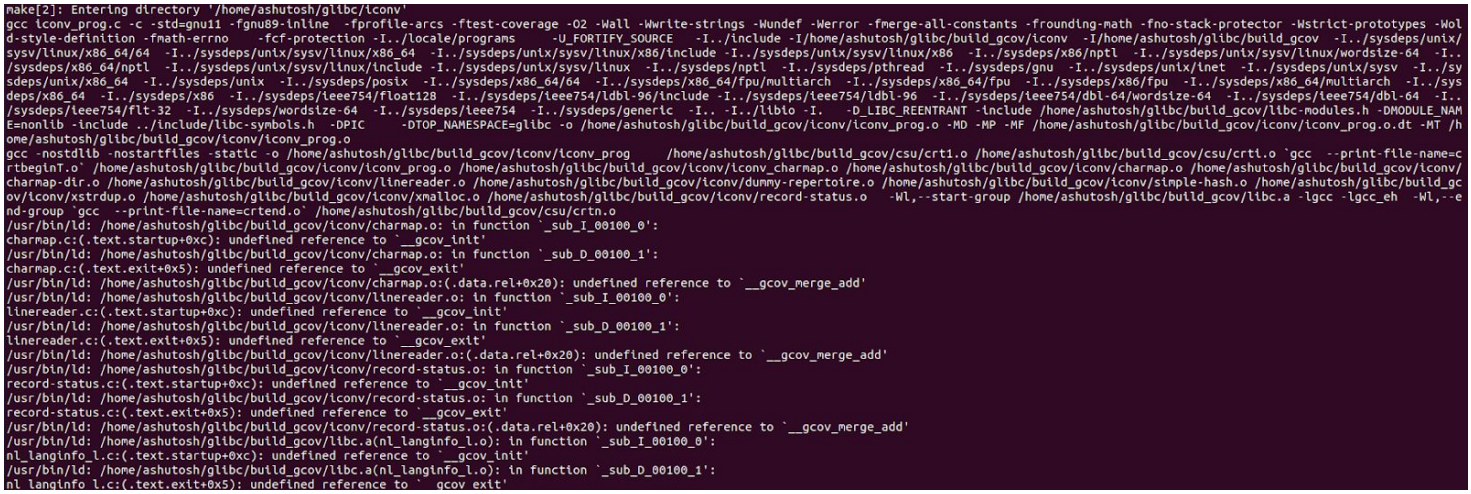
$ grep -ir fprofile-arcs .

```
ashutosh@ashu:~/glibc/build_gcov$ grep -ir fprofile-arcs .
./config.status:ac_cs_config="'--prefix=/home/ashutosh/glibc/install-gcov' '--disable-shared' 'CFLAGS=-fprofile-arcs -ftest-coverage -O2'"
./config.status:  set X /bin/bash '../configure'  '--prefix=/home/ashutosh/glibc/install-gcov' '--disable-shared' 'CFLAGS=-fprofile-arcs -ftest-coverage -O2' $ac_configure_extra_args --no-create --no-rec
ursion
./config.status:S["CFLAGS"]="-fprofile-arcs -ftest-coverage -O2"
./config.log:  $ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS=-fprofile-arcs -ftest-coverage -O2
./config.log:configure:2625: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:2673: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:3754: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:27: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:3879: gcc -fprofile-arcs -ftest-coverage -O2  -Werror -fstack-protector -xc /dev/null -S -o /dev/null
./config.log:configure:3899: gcc -fprofile-arcs -ftest-coverage -O2  -Werror -fstack-protector-strong -xc /dev/null -S -o /dev/null
./config.log:configure:3919: gcc -fprofile-arcs -ftest-coverage -O2  -Werror -fstack-protector-all -xc /dev/null -S -o /dev/null
./config.log:configure:4069: gcc -fprofile-arcs -ftest-coverage -O2   -nostartfiles -nostdlib -fno-stack-protector -pie -o conftest conftest.S
./config.log:configure:5111: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:5752: gcc -fprofile-arcs -ftest-coverage -O2
./config.log:configure:5793: gcc -fprofile-arcs -ftest-coverage -O2
./config.log:configure:5830: gcc -fprofile-arcs -ftest-coverage -O2   -fno-stack-protector
./config.log:configure:5863: gcc -fprofile-arcs -ftest-coverage -O2   -fno-stack-protector
./config.log:configure:5939: gcc -fprofile-arcs -ftest-coverage -O2   -fno-stack-protector
./config.log:configure:6031: gcc -fprofile-arcs -ftest-coverage -O2
./config.log:configure:6079: gcc -fprofile-arcs -ftest-coverage -O2  -S -fno-toplevel-reorder -fno-section-anchors
./config.log:configure:6114: gcc -fprofile-arcs -ftest-coverage -O2  -S -fPIC -mtls-dialect=gnu2
./config.log:configure:6285: gcc -fprofile-arcs -ftest-coverage -O2  -c conftest.c
./config.log:configure:6323: gcc -o conftest -fprofile-arcs -ftest-coverage -O2     conftest.c  -lgd -lpng -lz -lm >&5
./config.log:configure:6370: gcc -o conftest -fprofile-arcs -ftest-coverage -O2   conftest.c -lselinux   >&5
./config.log:configure:6521: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:6561: gcc -o conftest -fprofile-arcs -ftest-coverage -O2 -fPIC   -nostdlib -nostartfiles -shared -fno-stack-protector conftest.c  >&5
./config.log:configure:6604: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:28: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:87: gcc -c -fprofile-arcs -ftest-coverage -O2  conftest.c >&5
./config.log:configure:17: gcc -c -fprofile-arcs -ftest-coverage -O2 -fcf-protection -include cet.h conftest.c 1>&5
./config.log:configure:45: gcc -c -fprofile-arcs -ftest-coverage -O2 conftest.s -o conftest.o 1>&5
./config.log:ac_cv_env_CFLAGS_value='-fprofile-arcs -ftest-coverage -O2'
./config.log:CFLAGS='-fprofile-arcs -ftest-coverage -O2'
./config.make:CFLAGS = -fprofile-arcs -ftest-coverage -O2
```

$ make

```
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o): in function `_sub_I_00100_0':
dl-iteratephdr.c:(.text.startup+0xc): undefined reference to `__gcov_init'
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o): in function `_sub_D_00100_1':
dl-iteratephdr.c:(.text.exit+0x5): undefined reference to `__gcov_exit'
/usr/bin/ld: /home/ashutosh/glibc/build_gcov/libc.a(dl-iteratephdr.o):(.data.rel+0x20): undefined ref
erence to `__gcov_merge_add'
collect2: error: ld returned 1 exit status
make[2]: *** [../Rules:215: /home/ashutosh/glibc/build_gcov/iconv/iconv_prog] Error 1
make[2]: Leaving directory '/home/ashutosh/glibc/iconv'
make[1]: *** [Makefile:470: iconv/others] Error 2
make[1]: Leaving directory '/home/ashutosh/glibc'
make: *** [Makefile:9: all] Error 2
ashutosh@ashu:~/glibc/build_gcov$
```

$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared
$ make
$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS="-ftest-coverage -O2"
$ make
$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS="-fprofile-arcs -ftest-coverage -O2"
$ make
//make fails here



$ gcc -nostdlib -nostartfiles -static -o /home/ashutosh/glibc/build_gcov/iconv/iconvconfig /home/ashutosh/glibc/build_gcov/csu/crt1.o /home/ashutosh/glibc/build_gcov/csu/crti.o `gcc --print-file-name=crtbeginT.o` /home/ashutosh/glibc/build_gcov/iconv/iconvconfig.o /home/ashutosh/glibc/build_gcov/iconv/strtab.o /home/ashutosh/glibc/build_gcov/iconv/xmalloc.o /home/ashutosh/glibc/build_gcov/iconv/hash-string.o -Wl,--start-group /home/ashutosh/glibc/build_gcov/libc.a -lgcc -lgcc_eh -Wl,--end-group `gcc --print-file-name=crtend.o` /home/ashutosh/glibc/build_gcov/csu/crtn.o

$ gcc -nostdlib -nostartfiles -static -o /home/ashutosh/glibc/build_gcov/iconv/iconvconfig /home/ashutosh/glibc/build_gcov/csu/crt1.o /home/ashutosh/glibc/build_gcov/csu/crti.o `gcc --print-file-name=crtbeginT.o` /home/ashutosh/glibc/build_gcov/iconv/iconvconfig.o /home/ashutosh/glibc/build_gcov/iconv/strtab.o /home/ashutosh/glibc/build_gcov/iconv/xmalloc.o /home/ashutosh/glibc/build_gcov/iconv/hash-string.o -Wl,--start-group /home/ashutosh/glibc/build_gcov/libc.a -lgcc -lgcc_eh -Wl,--end-group `gcc --print-file-name=crtend.o` /home/ashutosh/glibc/build_gcov/csu/crtn.o -lgcov

//can’t find libgcov.a , so we add it manually.

//Passes successfully

$ ls

$ ./iconv/iconvconfig

$ ./iconv

$ cd iconv

$ ls

```
ashutosh@ashu:~/glibc/build_gcov/iconv$ ls
charmap-dir.gcno        gconv_conf.gcda     gconv_trans.gcda    iconv.os.dt
charmap-dir.o           gconv_conf.gcno     gconv_trans.gcno    iconv_prog
charmap-dir.o.d         gconv_conf.o        gconv_trans.o       iconv_prog.gcno
charmap-dir.o.dt        gconv_conf.o.d      gconv_trans.o.d     iconv_prog.o
charmap.gcno            gconv_conf.os       gconv_trans.os      iconv_prog.o.d
charmap.o               gconv_conf.os.dt    gconv_trans.os.dt   linereader.gcno
charmap.o.d             gconv_db.gcda       hash-string.gcda    linereader.o
charmap.o.dt            gconv_db.gcno       hash-string.gcno    linereader.o.d
dummy-repertoire.gcno   gconv_db.o          hash-string.o       linereader.o.dt
dummy-repertoire.o      gconv_db.o.d        hash-string.o.d     record-status.gcno
dummy-repertoire.o.d    gconv_db.os         hash-string.o.dt    record-status.o
dummy-repertoire.o.dt   gconv_db.os.dt      iconv_charmap.gcno  record-status.o.d
error.h.gcov            gconv_dl.gcda       iconv_charmap.o     record-status.o.dt
gconv_builtin.gcda      gconv_dl.gcno       iconv_charmap.o.d   simple-hash.gcno
gconv_builtin.gcno      gconv_dl.o          iconv_charmap.o.dt  simple-hash.o
gconv_builtin.o         gconv_dl.o.d        iconv_close.gcno    simple-hash.o.d
gconv_builtin.o.d       gconv_dl.os         iconv_close.o       simple-hash.o.dt
gconv_builtin.os        gconv_dl.os.dt      iconv_close.o.d     stamp.o
gconv_builtin.os.dt     gconv.gcda          iconv_close.os      stamp.os
gconv_cache.gcda        gconv.gcno          iconv_close.os.dt   stamp.oS
gconv_cache.gcno        gconv.o             iconvconfig         stdio.h.gcov
gconv_cache.o           gconv.o.d           iconvconfig.c.gcov  strtab.gcda
gconv_cache.o.d         gconv_open.gcda     iconvconfig.gcda    strtab.gcno
gconv_cache.os          gconv_open.gcno     iconvconfig.gcno    strtab.o
gconv_cache.os.dt       gconv_open.o        iconvconfig.o       strtab.o.d
gconv_charset.gcno      gconv_open.o.d      iconvconfig.o.d     strtab.o.dt
gconv_charset.o         gconv_open.os       iconvconfig.o.dt    xmalloc.c.gcov
gconv_charset.o.d       gconv_open.os.dt    iconv.gcno          xmalloc.gcda
gconv_charset.os        gconv.os            iconv.o             xmalloc.gcno
gconv_charset.os.dt     gconv.os.dt         iconv.o.d           xmalloc.o
gconv_close.gcda        gconv_simple.gcda   iconv_open.gcno     xmalloc.o.d
gconv_close.gcno        gconv_simple.gcno   iconv_open.o        xmalloc.o.dt
gconv_close.o           gconv_simple.o      iconv_open.o.d      xstrdup.gcno
gconv_close.o.d         gconv_simple.o.d    iconv_open.os       xstrdup.o
gconv_close.os          gconv_simple.os     iconv_open.os.dt    xstrdup.o.d
gconv_close.os.dt       gconv_simple.os.dt  iconv.os            xstrdup.o.dt
ashutosh@ashu:~/glibc/build_gcov/iconv$
```

$ gcov iconvconfig

```
ashutosh@ashu:~/glibc/build_gcov/iconv$ gcov iconvconfig
File 'iconvconfig.c'
Lines executed:41.79% of 390
Creating 'iconvconfig.c.gcov'
Cannot open source file iconvconfig.c

File '../include/bits/../../misc/bits/error.h'
Lines executed:50.00% of 2
Creating 'error.h.gcov'
Cannot open source file ../include/bits/../../misc/bits/error.h

File '../libio/bits/stdio.h'
Lines executed:0.00% of 1
Creating 'stdio.h.gcov'
Cannot open source file ../libio/bits/stdio.h
```

$ ls

$ vi iconvconfig.c.gcov

```
-:    0:Source:iconvconfig.c
-:    0:Graph:iconvconfig.gcno
-:    0:Data:iconvconfig.gcda
-:    0:Runs:1
```

$ ls *gcda

```
ashutosh@ashu:~/glibc/build_gcov/iconv$ ls *gcda
gconv_builtin.gcda  gconv_conf.gcda  gconv.gcda         gconv_trans.gcda  strtab.gcda
gconv_cache.gcda    gconv_db.gcda    gconv_open.gcda    hash-string.gcda  xmalloc.gcda
gconv_close.gcda    gconv_dl.gcda    gconv_simple.gcda  iconvconfig.gcda
ashutosh@ashu:~/glibc/build_gcov/iconv$
```

$ gcov xmalloc.gcda

```
ashutosh@ashu:~/glibc/build_gcov/iconv$ gcov xmalloc.gcda
File '../locale/programs/xmalloc.c'
Lines executed:50.00% of 22
Creating 'xmalloc.c.gcov'
Cannot open source file ../locale/programs/xmalloc.c

File '../include/bits/../../misc/bits/error.h'
Lines executed:0.00% of 4
Creating 'error.h.gcov'
Cannot open source file ../include/bits/../../misc/bits/error.h

ashutosh@ashu:~/glibc/build_gcov/iconv$
```

$ nano xmalloc.c.gcov

$ gcov -b xmalloc.gcda

$ nano xmalloc.c.gcov

$ gcov -a xmalloc.gcda

$ nano xmalloc.c.gcov

$ cd ..

$ ls

$ gcov iconv/xmalloc.c

$ ll

$ ls

$ nano xmalloc.c.gcov

$ find malloc.

$ find . -name ‘*.gcov’

```
ashutosh@ashu:~/glibc/build_gcov$ find . -name '*.gcov'
./iconv/iconvconfig.c.gcov
./iconv/error.h.gcov
./iconv/xmalloc.c.gcov
./iconv/stdio.h.gcov
./error.h.gcov
./xmalloc.c.gcov
```

$ cd ..

$ find . -name ‘*.gcov’

```
ashutosh@ashu:~/glibc$ find . -name '*.gcov'
./malloc.c.gcov
./hooks.c.gcov
./arena.c.gcov
./not-cancel.h.gcov
./malloc-sysdep.h.gcov
./build_gcov/iconv/iconvconfig.c.gcov
./build_gcov/iconv/error.h.gcov
./build_gcov/iconv/xmalloc.c.gcov
./build_gcov/iconv/stdio.h.gcov
./build_gcov/error.h.gcov
./build_gcov/xmalloc.c.gcov
./lowlevellock.h.gcov
ashutosh@ashu:~/glibc$
```

$ find . -name ‘*.gcno’

$ find . -name ‘*.gcno’ | grep malloc

```
ashutosh@ashu:~/glibc$ find . -name '*.gcno' | grep malloc
./build_gcov/iconv/xmalloc.gcno
./build_gcov/locale/xmalloc.gcno
./build_gcov/catgets/xmalloc.gcno
./build_gcov/support/xmalloc.gcno
./build_gcov/malloc/obstack.gcno
./build_gcov/malloc/set-freeres.gcno
./build_gcov/malloc/malloc.gcno
./build_gcov/malloc/scratch_buffer_grow.gcno
./build_gcov/malloc/alloc_buffer_create_failure.gcno
./build_gcov/malloc/mtrace.gcno
./build_gcov/malloc/dynarray_resize_clear.gcno
./build_gcov/malloc/alloc_buffer_copy_string.gcno
./build_gcov/malloc/scratch_buffer_grow_preserve.gcno
./build_gcov/malloc/mcheck-init.gcno
./build_gcov/malloc/dynarray_resize.gcno
./build_gcov/malloc/thread-freeres.gcno
./build_gcov/malloc/scratch_buffer_set_array_size.gcno
./build_gcov/malloc/dynarray_finalize.gcno
./build_gcov/malloc/alloc_buffer_alloc_array.gcno
./build_gcov/malloc/alloc_buffer_copy_bytes.gcno
./build_gcov/malloc/mcheck.gcno
./build_gcov/malloc/dynarray_at_failure.gcno
./build_gcov/malloc/alloc_buffer_allocate.gcno
./build_gcov/malloc/morecore.gcno
./build_gcov/malloc/dynarray_emplace_enlarge.gcno
./build_gcov/malloc/reallocarray.gcno
ashutosh@ashu:~/glibc$
```

$ gcov malloc/malloc.c

$ gcov build_gcov/malloc/malloc.c

```
ashutosh@ashu:~/glibc$ gcov build_gcov/malloc/malloc.c
File 'malloc.c'
Lines executed:16.95% of 1115
Creating 'malloc.c.gcov'
Cannot open source file malloc.c

File '../sysdeps/unix/sysv/linux/x86/lowlevellock.h'
Lines executed:100.00% of 2
Creating 'lowlevellock.h.gcov'
Cannot open source file ../sysdeps/unix/sysv/linux/x86/lowlevellock.h

File 'hooks.c'
Lines executed:2.56% of 156
Creating 'hooks.c.gcov'
Cannot open source file hooks.c

File 'arena.c'
Lines executed:6.79% of 265
Creating 'arena.c.gcov'
Cannot open source file arena.c

File '../sysdeps/unix/sysv/linux/malloc-sysdep.h'
Lines executed:0.00% of 10
Creating 'malloc-sysdep.h.gcov'
Cannot open source file ../sysdeps/unix/sysv/linux/malloc-sysdep.h

File '../sysdeps/unix/sysv/linux/not-cancel.h'
Lines executed:0.00% of 1
Creating 'not-cancel.h.gcov'
Cannot open source file ../sysdeps/unix/sysv/linux/not-cancel.h
```

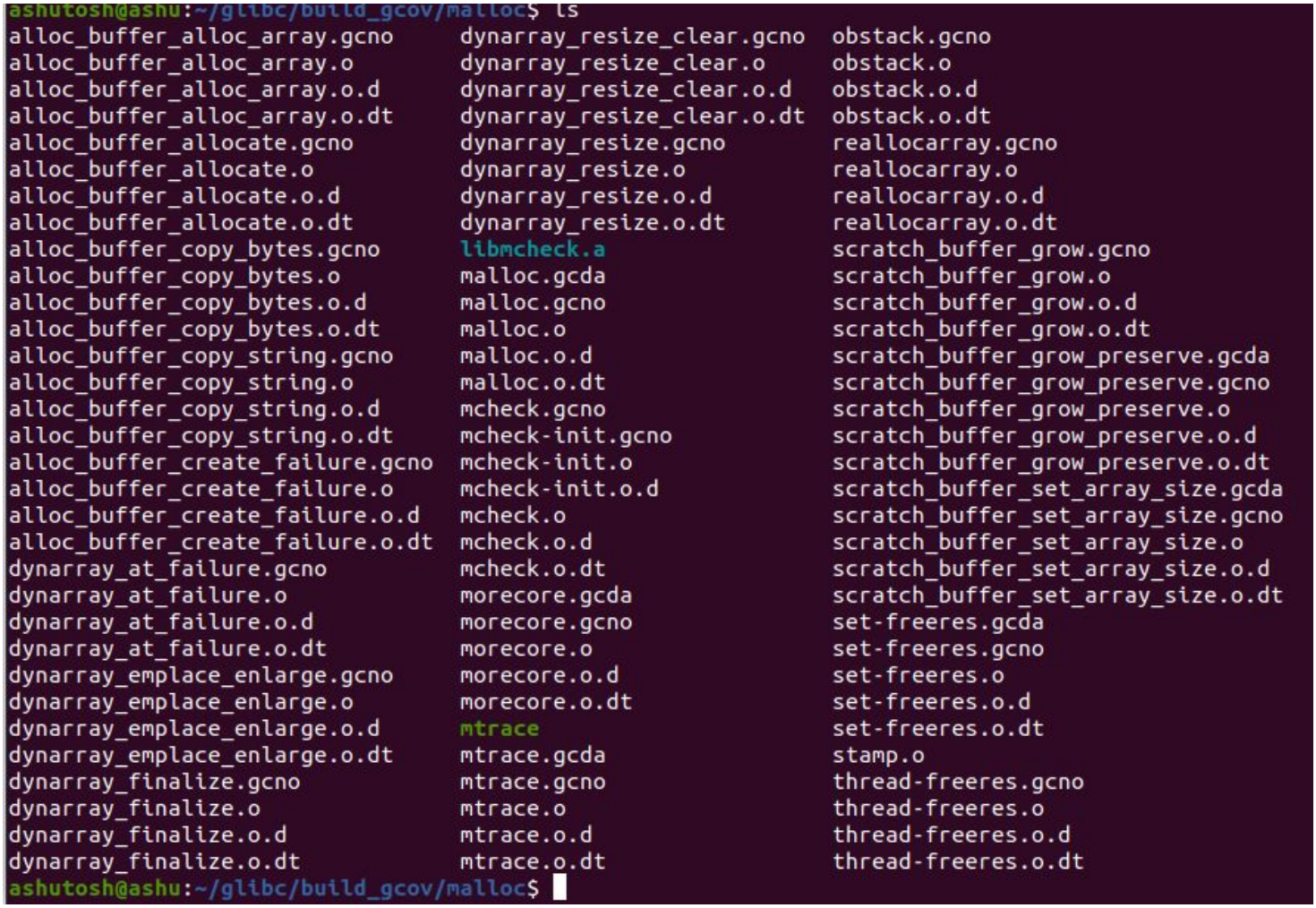
$ cd build_gcov

$ ls

$ nano malloc/malloc.c.gcov

$ cd malloc

$ ls



$ cd ..

$ find . -name malloc.c.gcov

$ cd ..

$ find . -name malloc.c.gcov

$ nano ./malloc.c.gcov

$ gcov build_gcov/malloc/malloc.c

$ man gcov

GCC 10.2.0

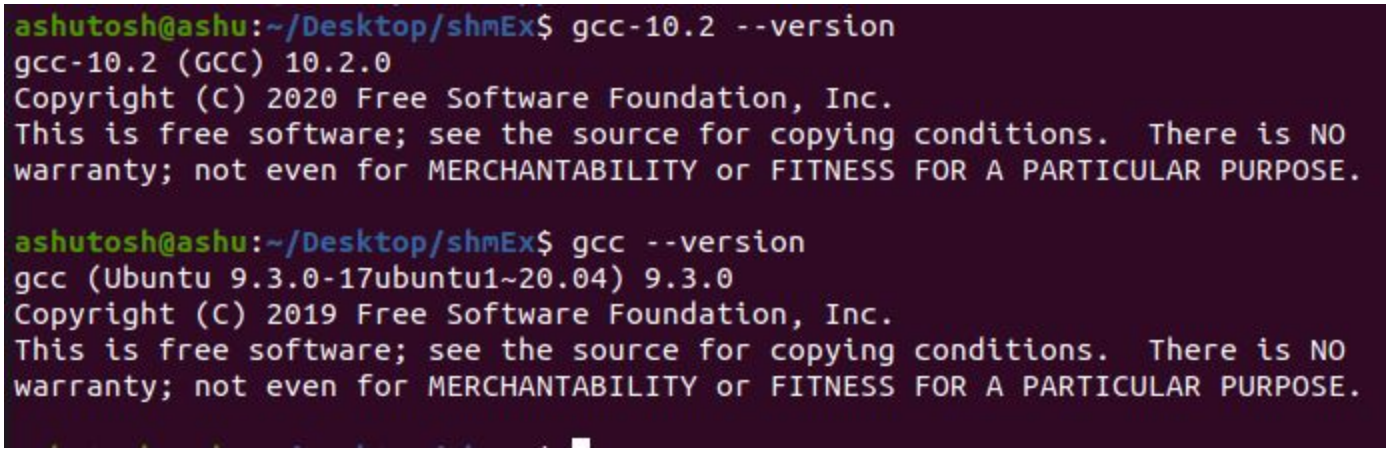
- Trying with -O0 and -Og flags didn't do anything.

It is important to have an alternate version of GCC installed so that I can make edits to GCC/GLibC without breaking the system.

https://gcc.gnu.org/install/finalinstall.html
https://solarianprogrammer.com/2016/10/07/building-gcc-ubuntu-linux/

To selectively invoke our alternate GCC, we can use gcc-10.2 instead of the default gcc option. This can be verified by:
gcc-10.2 --version //returns new GCC version -10.2.0

gcc --version //returns system gcc version -9.3.0



Any command that starts with gcc can be substituted with gcc-10.2 to invoke our new compiler.

To build our own GCC, we use the following commands:

$ sudo apt install build-essential wget m4 flex bison //prerequisite for GCC 10

//get the tarball
$ cd ~
$ wget https://ftpmirror.gnu.org/gcc/gcc-10.2.0/gcc-10.2.0.tar.xz
$ tar xf gcc-10.2.0.tar.xz
$ cd gcc-10.2.0
$ contrib/download_prerequisites

$ cd ~
$ mkdir build && cd build

$ ../gcc-10.2.0/configure -v --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu --prefix=/usr/local/gcc-10.2.0 --enable-checking=release --enable-languages=c,c++,fortran --disable-multilib --program-suffix=-10.2
$ make -j `nproc`

$ sudo make install-strip

To .bashrc add the following:

export export PATH=/usr/local/gcc-10.1.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/gcc-10.1.0/lib64:$LD_LIBRARY_PATH

. .bashrc

We are done installing alternate gcc

Rerunning the commands executed above shows that the glibc errors have not been patched since 9.3.0.

Downgrade to GCC 7.5

```
ashutosh@ashu:~$ gcc --version
gcc (Ubuntu 7.5.0-6ubuntu2) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ashutosh@ashu:~$ sudo update-alternatives --config gcc
There is only one alternative in link group gcc (providing /usr/bin/gcc): /usr/bin/gcc-7
Nothing to configure.
```

The configure scripts did not run:
$ ../configure --prefix=/home/ashutosh/glibc/build_gcov/install --disable-shared CFLAGS='-fprofile-arcs -ftest-coverage -O2'

```
configure: WARNING:
*** These auxiliary programs are missing or incompatible versions: msgfmt makeinfo
*** some features or tests will be disabled.
*** Check the INSTALL file for required versions.
checking LD_LIBRARY_PATH variable... contains current directory
configure: error:
*** LD_LIBRARY_PATH shouldn't contain the current directory when
*** building glibc. Please change the environment variable
*** and run configure again.
ashutosh@ashu:~/glibc/build_gcov$ make
make: *** No targets specified and no makefile found.  Stop.
ashutosh@ashu:~/glibc/build_gcov$
```

Even removing disabled shared flags and fprofile arcs and ftest coverage didn’t work

```
checking version of python3... 3.8.5, ok
configure: WARNING:
*** These auxiliary programs are missing or incompatible versions: msgfmt makeinfo
*** some features or tests will be disabled.
*** Check the INSTALL file for required versions.
checking LD_LIBRARY_PATH variable... contains current directory
configure: error:
*** LD_LIBRARY_PATH shouldn't contain the current directory when
*** building glibc. Please change the environment variable
*** and run configure again.
ashutosh@ashu:~/glibc/build_gcov$
```

Sample program that calls malloc

I wrote a simple program that uses malloc to allocate space, to check for gcov I added a few conditional statements and a couple of for loops. The program:

- Takes an input n
- Uses malloc to dynamically allocate memory of size n
- Check if the memory has been allocated successfully
- If allocated, print array of n elements
- If not, print memory not allocated
- Ask the user if they want to continue
- End

Program:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
char cont;

do {

// This pointer will hold the  base address of the block created
int* ptr;
int n, i;

// Get the number of elements for the array
printf("Enter number of elements:\n");
```

```
scanf("%d", &n);

// Dynamically allocate memory using malloc()
ptr = (int*)malloc(n * sizeof(int));

// Check if the memory has been successfully
// allocated by malloc or not
if (ptr == NULL) {
    printf("Memory not allocated.\n");
}
else {

    // Memory has been successfully allocated
    printf("Memory successfully allocated using malloc.\n");

    // Get the elements of the array
    for (i = 0; i < n; ++i) {
            ptr[i] = i + 1;
    }

    // Print the elements of the array
    printf("The elements of the array are: \n ");
    for (i = 0; i < n; ++i) {
            printf("%d, ", ptr[i]);
    }
          }
fflush(stdin);
printf("\n continue? (y/n) \n");
scanf(" %c", &cont);
}
while(cont=='y');
    return 0;
}
```

Running the program with coverage:

```
gcc -fprofile-arcs -ftest-coverage mallocExample.c
./a.out
```

Testcase Inputs:

```
Enter number of elements:
4
Memory successfully allocated using malloc.
```

The elements of the array are:
1, 2, 3, 4,

Enter number of elements:
654765978345435
Memory not allocated.
libgcov profiling error:/home/ashutosh/glibc/mallocEx/mallocExample.gcda:overwriting an existing profile data with a different timestamp

Error corrected when I deleted the previous gcno and gcda files from the mallocEx folder

```
ashutosh@ashu:~/glibc/mallocEx$ gcc -fprofile-arcs -ftest-coverage mallocExample.c
ashutosh@ashu:~/glibc/mallocEx$ ./a.out
Enter number of elements:
4
Memory successfully allocated using malloc.
The elements of the array are:
 1, 2, 3, 4,
 continue? (y/n)
y
Enter number of elements:
654765978345435
Memory not allocated.
libgcov profiling error:/home/ashutosh/glibc/mallocEx/mallocExample.gcda:overwriting an existing profile data with a different timestamp
ashutosh@ashu:~/glibc/mallocEx$ gcc -fprofile-arcs -ftest-coverage mallocExample.c
ashutosh@ashu:~/glibc/mallocEx$ ./a.out
Enter number of elements:
4
Memory successfully allocated using malloc.
The elements of the array are:
 1, 2, 3, 4,
 continue? (y/n)
y
Enter number of elements:
654765978345435
Memory not allocated.
ashutosh@ashu:~/glibc/mallocEx$
```

Trying with more inputs:

3
y

32743608576328965803265083865092 3
y

12
y
3
y

34327638062073943287
n

```
ashutosh@ashu:~/glibc/mallocEx$ gcc -fprofile-arcs -ftest-coverage mallocExample.c
ashutosh@ashu:~/glibc/mallocEx$ ./a.out
Enter number of elements:
3
Memory successfully allocated using malloc.
The elements of the array are:
 1, 2, 3,
 continue? (y/n)
y
Enter number of elements:
327436085763289658032650838650923
Memory not allocated.

 continue? (y/n)
y
Enter number of elements:
12
Memory successfully allocated using malloc.
The elements of the array are:
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
 continue? (y/n)
y
Enter number of elements:
3
Memory successfully allocated using malloc.
The elements of the array are:
 1, 2, 3,
 continue? (y/n)
y
Enter number of elements:
34327638062073943287
Memory not allocated.

 continue? (y/n)
n
```

gcov mallocExample.c

We can see the mallocExample.gcda and mallocExample.gcno files are generated after the program is done running.

```
ashutosh@ashu:~/glibc/mallocEx$ gcov mallocExample.c
File 'mallocExample.c'
Lines executed:100.00% of 17
Creating 'mallocExample.c.gcov'

ashutosh@ashu:~/glibc/mallocEx$ ls
a.out  mallocExample  mallocExample.c  mallocExample.c.gcov  mallocExample.gcda  mallocExample.gcno
```

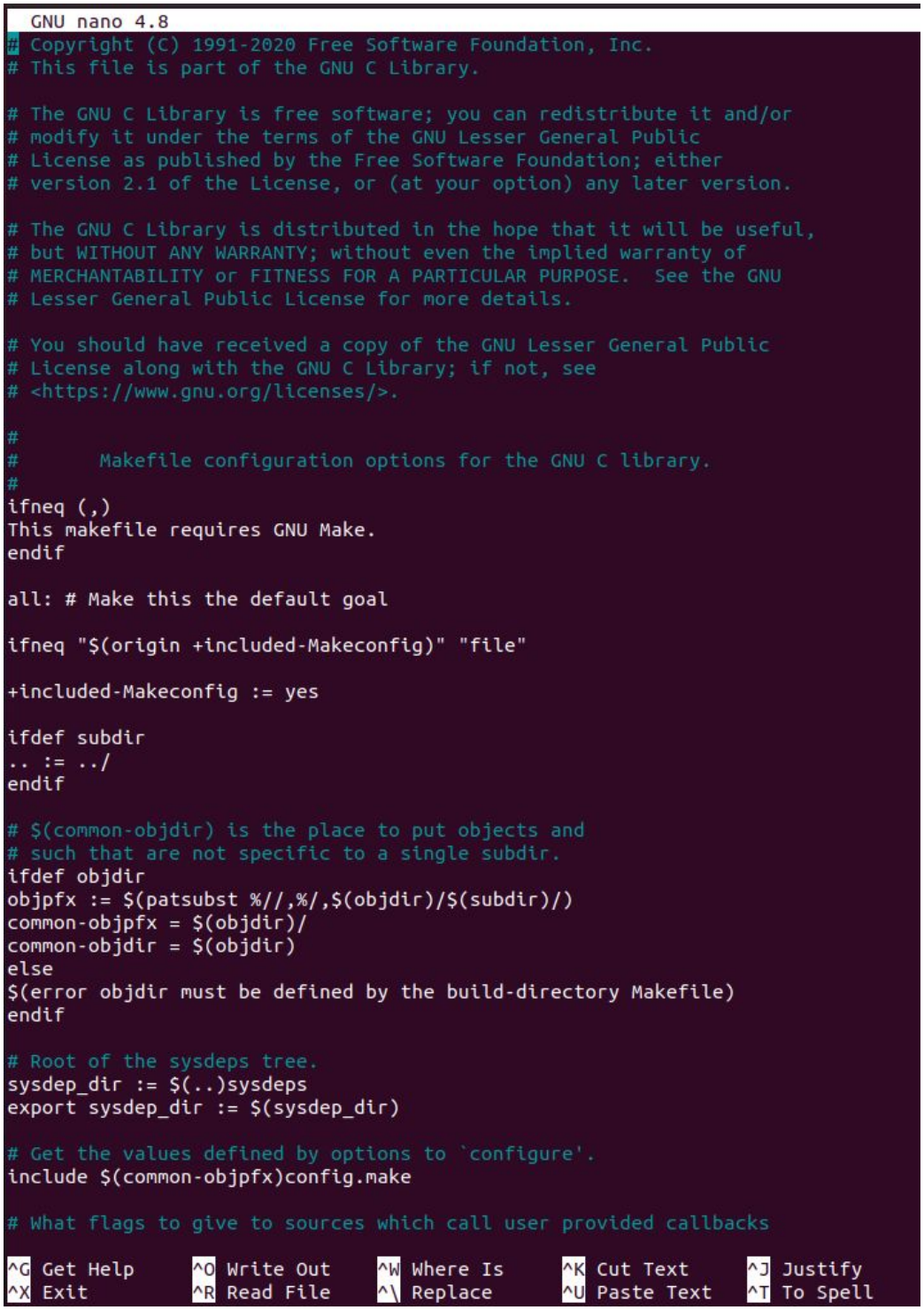
```
        -:    0:Source:mallocExample.c
        -:    0:Graph:mallocExample.gcno
        -:    0:Data:mallocExample.gcda
        -:    0:Runs:1
        -:    1:#include <stdio.h>
        -:    2:#include <stdlib.h>
        -:    3:
        5:    4:int main()
        -:    5:{
        -:    6:char cont;
        -:    7:
        -:    8:do {
        -:    9:
        -:   10:// This pointer will hold the  base address of the block created
        -:   11:int* ptr;
        -:   12:int n, i;
        -:   13:
        -:   14:// Get the number of elements for the array
        5:   15:printf("Enter number of elements:\n");
        5:   16:scanf("%d", &n);
        -:   17:
        -:   18:// Dynamically allocate memory using malloc()
        5:   19:ptr = (int*)malloc(n * sizeof(int));
        -:   20:
        -:   21:// Check if the memory has been successfully
        -:   22:// allocated by malloc or not
        5:   23:if (ptr == NULL) {
        2:   24:       printf("Memory not allocated.\n");
        -:   25:}
        -:   26:else {
        -:   27:
        -:   28:       // Memory has been successfully allocated
        3:   29:       printf("Memory successfully allocated using malloc.\n");
        -:   30:
        -:   31:       // Get the elements of the array
       21:   32:       for (i = 0; i < n; ++i) {
       18:   33:               ptr[i] = i + 1;
        -:   34:       }
        -:   35:
        -:   36:       // Print the elements of the array
        3:   37:       printf("The elements of the array are: \n ");
       21:   38:       for (i = 0; i < n; ++i) {
       18:   39:               printf("%d, ", ptr[i]);
        -:   40:       }
        -:   41:    }
        5:   42:fflush(stdin);
        5:   43:printf("\n continue? (y/n) \n");
        5:   44:scanf(" %c", &cont);
        -:   45:}
        5:   46:while(cont=='y');
        1:   47:       return 0;
        -:   48:}
```

Tried the following commands after patching gcc

$ ../configure --prefix=/home/ashutosh/glibc/build_gcov/install --disable-shared CFLAGS='-fprofile-arcs -ftest-coverage -O2'
$ make

//fails libc-modules.h not found

$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS="-fprofile-arcs -ftest-coverage -O2"



Edited a few lines.

$ ../configure --prefix=/home/ashutosh/glibc/install-gcov --disable-shared CFLAGS="-fprofile-arcs -ftest-coverage -O2" --disabled-nscd

Tried make -all and make -k install

```
#!/bin/sh


rm -f test1.o

rm -f test1


gcc -c -g test1.c -fprofile-arcs -ftest-coverage -I../install-gcov/include  -o test1.o


# -fprofile-args not needed in link stage

#gcc -nostdlib -nostartfiles -o test2-toupper -fprofile-arcs -ftest-coverage -Wl,-z,combreloc -Wl,-z,relro -Wl,--hash-style=both \

gcc -nostdlib -nostartfiles -o test1 \

        ../install-gcov/lib/crti.o \

        ../install-gcov/lib/crt1.o \

        `gcc  --print-file-name=crtbeginS.o` \

        ./test1.o \

        -L../install-gcov/lib \

        `gcc  --print-file-name=crtendS.o` \

        ../install-gcov/lib/crtn.o \

        -lgcov -lc -lgcc_eh -lc -lgcc
```

The test program:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
	int *ptr;
	int i;

	ptr = (int *) malloc(10 * sizeof(int));

	for (i = 0; i < 10; i++)
		ptr[i] = i;

	printf("%d\n", ptr[1]);
	return 0;
}
```

https://thoughtbot.com/blog/the-magic-behind-configure-make-make-install

https://sourceware.org/glibc/wiki/Contribution%20checklist

https://www.gnu.org/licenses/why-assign.en.html

https://sourceware.org/glibc/wiki/Testing/Testsuite
Ask question on libc-help
Enable shared bug. Libc-modules.h not found
Learn about AWK, Makeconfig libc-modules.h ,Soversions.i
Trying to build in ELF Directory. Error because of non linking. Trying to link because symbols not found.

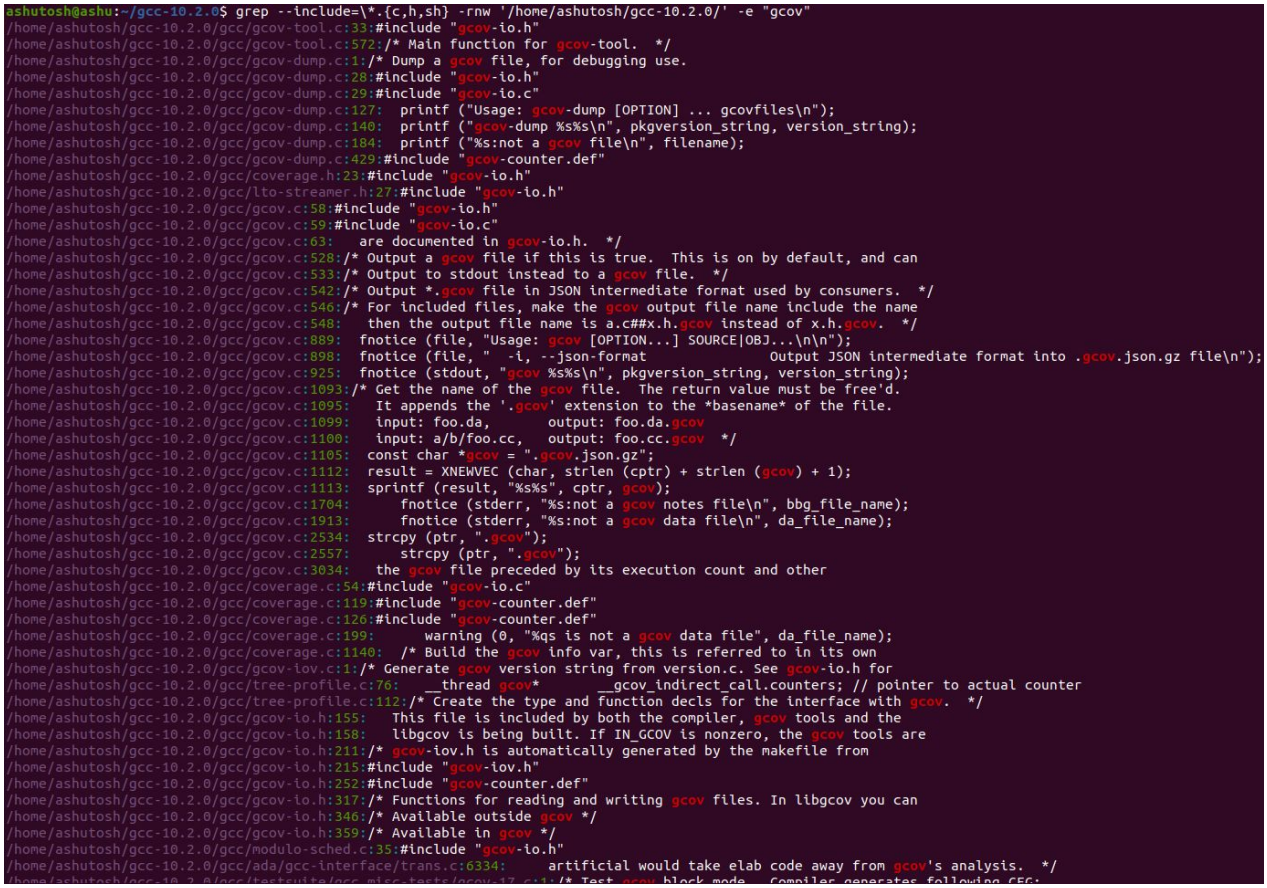
Trying to fix the error

$ grep --include=*.{c,h,sh,i,o} -rnw '/home/ashutosh/glibc/' -e "gcov"

Returns nothing

$ grep --include=*.{c,h,sh} -rnw '/home/ashutosh/gcc-10.2.0/' -e "gcov"

Has lots of references to gcov



Searching for -fprofile-arcs in gcc

$ grep --include=*.{c,h,sh,i,o} -rnw '/home/ashutosh/gcc-10.2.0/' -e "fprofile-arcs"

The static libraries.
link-libc-static = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib) -Wl,--end-group
link-libc-static-tests = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib-tests) -Wl,--end-group

Fixing the gcov build bug

We need to use the ../configure -disable-shared CFLAGS=”-fprofile-arcs -ftest-coverage” inorder to compile glibc with coverage.

But since this doesn’t work, we need to include a new flag in glibc so that we can perform code coverage of glibc.

Using the configure.ac script (autoconf) we can define a new flag using AC_ARG_ENABLE

From the autoconf manual, page 285

https://autotools.io/autoconf/arguments.html

`AC_ARG_ENABLE` (*feature*, *help-string*, [*action-if-given*], [*action-if-not-given*]) [Macro]

If the user gave `configure` the option '`--enable-`*feature*' or '`--disable-`*feature*', run shell commands *action-if-given*. If neither option was given, run shell commands *action-if-not-given*. The name *feature* indicates an optional user-level facility. It should consist only of alphanumeric characters, dashes, plus signs, and dots.

The option's argument is available to the shell commands *action-if-given* in the shell variable `enableval`, which is actually just the value of the shell variable named `enable_`*feature*, with any non-alphanumeric characters in *feature* changed into '_'. You may use that variable instead, if you wish. The *help-string* argument is like that of `AC_ARG_WITH` (see Section 15.2 [External Software], page 273).

You should format your *help-string* with the macro `AS_HELP_STRING` (see Section 15.4 [Pretty Help Strings], page 276).

See the examples suggested with the definition of `AC_ARG_WITH` (see Section 15.2 [External Software], page 273) to get an idea of possible applications of `AC_ARG_ENABLE`.

https://www.gnu.org/software/autoconf/manual/autoconf-2.69/html_node/Package-Options.html

https://autotools.io/autoconf/arguments.html

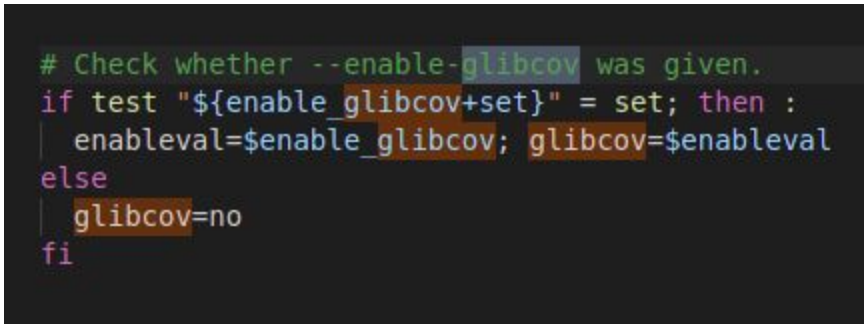
https://www.phoronix.com/scan.php?page=news_item&px=glibc-malloc-thread-cache

https://www.gnu.org/prep/standards/html_node/Configuration.html

https://stackoverflow.com/questions/2531827/what-are-makefile-am-and-makefile-in

Adding the options to the configure.ac file and running:
$autoconf configure.ac
$autoreconf configure.ac

```
AC_ARG_ENABLE([glibcov],
              AC_HELP_STRING([--enable-glibcov],
                 [enable code coverage for glibc]),
              [glibcov=$enableval],
              [glibcov=no])
AC_SUBST(glibcov)
```



```
# Check whether --enable-glibcov was given.
if test "${enable_glibcov+set}" = set; then :
  enableval=$enable_glibcov; glibcov=$enableval
else
  glibcov=no
fi
```

Now we need to make an edit to configparms.

Starting from a clean version 2.32 do the following.

Then please prepare the patch. Note that we have to patch additional files – the README/INSTALL, other help files. We have to explain the procedure in the README file (first build a shared lib to get the libc-modules.h file, then build with the configure command that I sent you before. Also note that the make command must be: “make CXX=”

Steps to patch GlibC

Makeconfig:

Add the following lines at line 612 (location is important!)

#

support for code coverage with gcov

#

ifeq (yes,$(build-gcov))

+cflags += -fprofile-arcs -ftest-coverage -O2

link-libc-static = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib) -Wl,--end-group

link-libc-static-tests = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib-tests) -Wl,--end-group

endif

config.make.in:

Add in line 95:

build-gcov = @gcov@

configure:

Add in line 597:

gcov

Add in line: 771:

enable_gcov

add in line 1423:

--enable-coverage build with instrumentation for gcov code coverage
library [default=no]

Add in line 3381:

```
# Check whether --enable-gcov was given.
if test "${enable_gcov+set}" = set; then :
  enableval=$enable_coverage; gcov=$enableval
else
  gcov=no
fi
```

configure.ac:

add in line 178:

```
AC_ARG_ENABLE([gcov],

           AC_HELP_STRING([--enable-gcov],

                          [build with instrumentation for gcov code coverage
library @<:@default=no@:>@]),

           [gcov=$enableval],

           [gcov=no])

AC_SUBST(gcov)
```

Final Patchfile

```
diff --git a/INSTALL b/INSTALL
index 41f5d6b708..19ee1f7568 100644
--- a/INSTALL
+++ b/INSTALL
@@ -111,6 +111,21 @@ if 'CFLAGS' is specified it must enable optimization.  For example:
         systems support shared libraries; you need ELF support and
         (currently) the GNU linker.

+'--enable-gcov'
+       Enable building glibc with instrumentation for code coverage. For
+       individual files '--coverage' can be used, but this fails
+       when it is used as an option with '../configure' command to build glibc.
+       To successfully build glibc with coverage, you must first build glibc
+       with shared libraries enabled. This will allow some files such as
+       'libc-modules.h' to be generated which are needed for code coverage to
+       work.
+       The command '../configure --disable-shared --enable-gcov --disable-shared
+       --without-selinux --disable-nscd --prefix=/path/to/install'
```

```
+       will build glibc with code coverage. Note that not including a prefix,
+       or setting the prefix to usr/lib can affect the system glibc and can
+       render the system unusable. The command to make is 'make CXX=', without
+       which you will get a error 'cannot find -lgcc_s'.
+
 '--enable-static-pie'
        Enable static position independent executable (static PIE) support.
        Static PIE is similar to static executable, but can be loaded at
diff --git a/Makeconfig b/Makeconfig
index dfda418aac..e78d334b35 100644
--- a/Makeconfig
+++ b/Makeconfig
@@ -610,6 +610,13 @@ endif
 link-libc-static = -Wl,--start-group $(common-objpfx)libc.a $(static-gnulib) -Wl,--end-group
 link-libc-static-tests = -Wl,--start-group $(common-objpfx)libc.a $(static-gnulib-tests) -Wl,--end-group

+# support for code coverage with gcov
+ifeq (yes,$(build-gcov))
++cflags += -fprofile-arcs -ftest-coverage -O2
+link-libc-static = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib) -Wl,--end-group
+link-libc-static-tests = -Wl,--start-group -lgcov $(common-objpfx)libc.a $(static-gnulib-tests) -Wl,--end-group
+endif
+
 # How to link against libgcc.  Some libgcc functions, such as those
 # for "long long" arithmetic or software floating point, can always be
 # built without use of C library headers and do not have any global
diff --git a/config.make.in b/config.make.in
index 1ac9417245..463e9d808c 100644
--- a/config.make.in
+++ b/config.make.in
@@ -92,6 +92,7 @@ build-shared = @shared@
 build-pic-default= @libc_cv_pic_default@
 build-pie-default= @libc_cv_pie_default@
 cc-pie-default= @libc_cv_cc_pie_default@
+build-gcov = @gcov@
 build-profile = @profile@
 build-static-nss = @static_nss@
 cross-compiling = @cross_compiling@
diff --git a/configure b/configure
index 4795e721e5..c0f35c9542 100755
--- a/configure
+++ b/configure
@@ -683,6 +683,7 @@ force_install
 bindnow
```

```
 hardcoded_path_in_tests
 enable_timezone_tools
+gcov
 extra_nonshared_cflags
 use_default_link
 sysheaders
@@ -731,6 +732,7 @@ infodir
 docdir
 oldincludedir
 includedir
+runstatedir
 localstatedir
 sharedstatedir
 sysconfdir
@@ -765,6 +767,7 @@ with_default_link
 with_nonshared_cflags
 enable_sanity_checks
 enable_shared
+enable_gcov
 enable_profile
 enable_static_pie
 enable_timezone_tools
@@ -842,6 +845,7 @@ datadir='${datarootdir}'
 sysconfdir='${prefix}/etc'
 sharedstatedir='${prefix}/com'
 localstatedir='${prefix}/var'
+runstatedir='${localstatedir}/run'
 includedir='${prefix}/include'
 oldincludedir='/usr/include'
 docdir='${datarootdir}/doc/${PACKAGE_TARNAME}'
@@ -1094,6 +1098,15 @@ do
   | -silent | --silent | --silen | --sile | --sil)
         silent=yes ;;

+  -runstatedir | --runstatedir | --runstatedi | --runstated \
+  | --runstate | --runstat | --runsta | --runst | --runs \
+  | --run | --ru | --r)
+        ac_prev=runstatedir ;;
+  -runstatedir=* | --runstatedir=* | --runstatedi=* | --runstated=* \
+  | --runstate=* | --runstat=* | --runsta=* | --runst=* | --runs=* \
+  | --run=* | --ru=* | --r=*)
+        runstatedir=$ac_optarg ;;
+
   -sbindir | --sbindir | --sbindi | --sbind | --sbin | --sbi | --sb)
```

```
          ac_prev=sbindir ;;
  -sbindir=* | --sbindir=* | --sbindi=* | --sbind=* | --sbin=* \
@@ -1231,7 +1244,7 @@ fi
 for ac_var in    exec_prefix prefix bindir sbindir libexecdir datarootdir \
                  datadir sysconfdir sharedstatedir localstatedir includedir \
                  oldincludedir docdir infodir htmldir dvidir pdfdir psdir \
-          libdir localedir mandir
+          libdir localedir mandir runstatedir
 do
   eval ac_val=\$$ac_var
   # Remove trailing slashes.
@@ -1384,6 +1397,7 @@ Fine tuning of the installation directories:
   --sysconfdir=DIR     read-only single-machine data [PREFIX/etc]
   --sharedstatedir=DIR          modifiable architecture-independent data [PREFIX/com]
   --localstatedir=DIR  modifiable single-machine data [PREFIX/var]
+  --runstatedir=DIR    modifiable per-process data [LOCALSTATEDIR/run]
   --libdir=DIR             object code libraries [EPREFIX/lib]
   --includedir=DIR     C header files [PREFIX/include]
   --oldincludedir=DIR  C header files for non-gcc [/usr/include]
@@ -1420,6 +1434,8 @@ Optional Features:
   --disable-sanity-checks really do not use threads (should not be used except
                    in special situations) [default=yes]
   --enable-shared      build shared library [default=yes if GNU ld]
+  --enable-gcov        build with instrumentation for gcov code coverage
+                       library [default=no]
   --enable-profile     build profiled library [default=no]
   --enable-static-pie  enable static PIE support and use it in the
                    testsuite [default=no]
@@ -3361,6 +3377,16 @@ else
   shared=yes
 fi

+
+# Check whether --enable-gcov was given.
+if test "${enable_gcov+set}" = set; then :
+  enableval=$enable_gcov; gcov=$enableval
+else
+  gcov=no
+fi
+
+
+
 # Check whether --enable-profile was given.
 if test "${enable_profile+set}" = set; then :
```

```
  enableval=$enable_profile; profile=$enableval
diff --git a/configure.ac b/configure.ac
index 93e68fb696..d5b27a4909 100644
--- a/configure.ac
+++ b/configure.ac
@@ -174,6 +174,14 @@ AC_ARG_ENABLE([shared],
                                  [build shared library @<:@default=yes if GNU ld@:>@]),
                [shared=$enableval],
                [shared=yes])
+
+AC_ARG_ENABLE([gcov],
+       AC_HELP_STRING([--enable-gcov],
+       [build with instrumentation for gcov code coverage library @<:@default=no@:>@]),
+       [gcov=$enableval],
+       [gcov=no])
+AC_SUBST(gcov)
+
 AC_ARG_ENABLE([profile],
               AC_HELP_STRING([--enable-profile],
                              [build profiled library @<:@default=no@:>@]),
diff --git a/manual/install.texi b/manual/install.texi
index 735e99bb03..5d10340d05 100644
--- a/manual/install.texi
+++ b/manual/install.texi
@@ -141,6 +141,21 @@ Don't build shared libraries even if it is possible.  Not all systems
 support shared libraries; you need ELF support and (currently) the GNU
 linker.

+@item --enable-gcov
+Enable building glibc with instrumentation for code coverage. For
+individual files @option{--coverage} can be used, but this fails
+when it is used as an option with @code{configure} command to build glibc.
+To successfully build glibc with coverage, you must first build glibc
+with shared libraries enabled. This will allow some files such as
+@file{libc-modules.h} to be generated which are needed for code coverage
+to work.
+The command @samp{../configure --disable-shared --enable-gcov
+--disable-shared --without-selinux --disable-nscd --prefix=/path/to/install}
+will build glibc with code coverage. Note that not including a prefix,
+or setting the prefix to usr/lib can affect the system glibc and can
+render the system unusable. The command to make is 'make CXX=', without
+which you will get a error 'cannot find -lgcc_s'.
+
 @item --enable-static-pie
```

Enable static position independent executable (static PIE) support.
Static PIE is similar to static executable, but can be loaded at any