

Forum:[Hack Forums](#)

Thread: [C++ compiler flags for function implementation arrangement in object file](#)

6577657534735 - June 20, 2019, 1:46 am

If I have func1() func2() and func3() and I want the compiler to arrange the implementations of func1() before func2() and func2() before func3() in-memory for instruction cache locality reasons, does anyone know if any compilers have flags to do this?

YouthInAsia - June 20, 2019, 3:42 am

The compiler should automatically do this as far as I'm aware but it might just be unspecified behavior. I do remember though marking functions with the "noinline" attribute seemed to keep them in place. So try just giving each function a noinline attribute and see if that does the trick. It might just vary compiler to compiler, but I honestly doubt anything like that would ever be standard.

So I tried this code

Code #include <stdio.h>

```
void __attribute__((noinline)) f1(){
    puts("in f1");
}
void __attribute__((noinline)) f2(){
    puts("in f2");
}
void __attribute__((noinline)) f3(){
    puts("in f3");
}
int main(){
    f3();
    f2();
    f1();
    puts("break");
}
```

Their placement in memory regardless of which order they were called was still f1 first, f2 second f3 third

```
Code 0x100000f48 <+8>: call 0x100000f20 ; f3 at main.cpp:9
0x100000f4d <+13>: call 0x100000f00 ; f2 at main.cpp:6
0x100000f52 <+18>: call 0x100000ee0 ; f1 at main.cpp:3
-> 0x100000f57 <+23>: lea rdi, [rip + 0x47] ; "break"
0x100000f5e <+30>: call 0x100000f70 ; symbol stub for: puts
```

So that does seem to work but when I remove the noinline attribute I get the same result

```
Code 0x100000f48 <+8>: call 0x100000f20 ; f3 at main.cpp:9
0x100000f4d <+13>: call 0x100000f00 ; f2 at main.cpp:6
0x100000f52 <+18>: call 0x100000ee0 ; f1 at main.cpp:3
-> 0x100000f57 <+23>: lea rdi, [rip + 0x47] ; "break"
```

Which means the code section should just get compiled in the order you wrote it like it did in my case. But I think if you want to be extra careful then yes place noinline attribute.

Another thing that could do the trick is marking a function as volatile/const.

6577657534735 - June 20, 2019, 9:35 am

Thanks, it's worrying that they could randomly swap order if you switched compiler versions or modified optimization flags or something, but I will look into whether this applies to me

YouthInAsia - June 20, 2019, 10:09 am

(Jun 20, 2019 09:35 AM)Abraluna Wrote: [\[CITE\]](#) Thanks, it's worrying that they could randomly swap order if you switched compiler versions or modified optimization flags or something, but I will look into whether this applies to me

Yes one things for certain, even with compiler optimisation turned on a function with noinline attribute will still be separated.

dracx - June 20, 2019, 6:08 pm

This gretly varies from compiler to compiler, plus I take note of the object files (even just of your headers), if your worried bout their placement too, you should try and research more. And judging from your subject, I suggest disabling ASLR and other protection mechanisms that makes your stuff go dynamic.

doxerhacker - June 20, 2019, 9:48 pm

other than forcing no inline you should also make sure incremental linking is disabled.