

了解你的编译器——基于 gcc 编译器实现

李伟 1711350 2017 级计算机一班

摘要

本文基于 gcc 编译器，对于 c++ 源程序编译为机器码的过程进行逐步的解析，分析编译器在每一个流程中所执行的具体的功能是什么，解构从 cpp 文件到 EXE 文件中间所经历的所有过程，了解编译器内部的工作流程和原理。同时在此基础上，进一步探索 gcc/g++ 编译器提供的调试和优化选项的使用及其效果影响，对生成的文件进行对比分析得出结论。

关键字： gcc/g++ 编译器 预处理 编译 汇编 优化 调试 代码生成

目录

一、引言	3
二、编译器发展历史	3
2.1 编译器发展简史 [1]	3
2.2 gcc/g++ 编译器发展史	4
三、实验内容及环境	4
3.1 实验内容	4
3.2 实验环境	5
四、编译器了解初探	6
4.1 预处理	6
4.2 编译	7
4.3 汇编	9
4.4 链接	11
4.5 总结	11
五、编译器知识进阶——调试与优化	12
5.1 调试	12
5.2 优化	13
六、结论	14
参考文献	15
附录 A 生成汇编代码 fib.s	16
附录 B gcc/g++ 链接 cpp 文件结果对比	20
附录 C 反汇编生成代码	22
附录 D O3-O0 优化汇编代码对比	26

一、引言

计算机运转的控制在于指令流的执行，经过了六十多年的发展，计算机已经有了翻天覆地的变化，体积不断缩小，运算能力飞速提高，随之进行的也是编程语言的更新换代。从最初的二进制语言到汇编语言，再到后来的 C/C++、Python 等高级语言，计算机语言越来越接近于人类的自然语言，而机器却难以理解，编译器便是连接高级语言和二进制机器码之间的桥梁，编译器通过预处理、编译、汇编、链接、代码优化、生成机器码等流程，将程序员编写的高级语言代码翻译转换为机器能够理解的二进制文件（可执行文件）。

编译器的存在大大提升了代码从业者的编程效率，降低了编程的难度，在很大程度上促进了计算机产业的发展，而理解编译器的工作原理也能够进一步提升编程人员的编程质量，提高程序的健壮性、可移植性及可维护性。

二、编译器发展历史

2.1 编译器发展简史 [1]

编译器为一种计算机程序，负责将源代码（原始语言）转换成另外一种目标语言，通过一系列的流程生成可执行文件交由计算机执行。编译器的发展紧随着计算机的飞速发展，也有了近五十年的发展历史 [3]。

20 世纪 60 年代，人们对计算机的运算速度要求不断提升，计算机架构师和编译器编写者开始考虑利用并行性提升运算速度，但是这也使得程序员面临着编程难度提升的难题，在这种需求的刺激下，编译器开始提升了分析的水平，协助编程人员提高编程效率。这期间，TI SAC 机器拥有了第一个自动向量化编译器¹。

20 世纪 70 年代，编译器分析技术和优化能力进一步提升，其中最重要的进步就在于能够为程序员提供编译器的反馈，能够提供程序员在那些代码位置进行了向量化优化等编译信息，这一附加功能进步提高了程序员的编程效率和代码维护的方便性。同时在 1975 年，Steve Johnson 为 Unix 系统编写的 Yacc 就是编译器自动构造（编译器的编译器）中的佼佼者，与之类似的是在有限状态自动机的研究带动下发展的一种扫描程序生成器，lex（由 Mike Lesk 为 Unix 系统开发）便是其中的佼佼者。

20 世纪 80 年代之前，多处理器已经问世，编译器编写者开始尝试自动并行化，然而在实际应用当中却没有取得很好的效果，而是退而求其次，让程序员的介入实现并行性。

20 世纪九十年代，openMP 开始大行其道，虽然取得了很大的成功，但是却没有体现出编译器的优化能力，在 2000 年之后。多核微处理器迅速发展，并行性编程问题亟

¹向量化指的是编译器能够对代码中的循环进行优化，重写循环，生成运行效率更高的矢量化代码

待解决，相应的针对于并行化编程的编译器也开始发展。但是近二十年以来，编译器的核心部分变动不大，编译器的相关知识也开始成为经典的教学科目。

2.2 gcc/g++ 编译器发展史

GCC/G++（GNU Compiler Collection）全称为 GNU 编译套装，1985 年由理查德·马修·斯托曼开始发展，现由自由软件基金会负责维护工作。起初原名为 GNU C 语言编译器，只能够处理 C 语言，之后逐渐拓展，能够处理 C++、Java、Python、Go 等多种语言。

GCC 在 1985 年一开始由理查德在旧有编译器的基础上利用 `pastel` 语言进行了拓展，使之能够支持 C 语言，1987 年由斯托曼和 Len Tower 使用 C 语言重写，并成为 GNU 目的的编译器。

1997 年，一部分不满 GCC 封闭创作环境的人，组织一个名为 EGCS 的项目，最终在 1999 年 4 月 EGCS 成为了 GCC 的官方版本。值得注意的是，GCC4.2 之前的版本，如果希望在 C、C++、Fortran 中嵌入 `openMP` 语句，需要额外安装一些库在 4.2 之后的版本开始支持 `openMP`。

三、实验内容及环境

3.1 实验内容

本实验要求以某一具体编译器为研究对象，深入地探究语言处理系统的完整工作过程，具体包含一下内容：

1. 预处理器做了什么？
2. 编译器做了什么？
3. 汇编器做了什么？
4. 链接器做了什么？

实验示例程序代码如下代码所示

```
#include<iostream>
using namespace std;

int main() {
    int a, b, i, t, n;
    a = 0;
    b = 1;
```

```

i = 1;
cin >> n;
cout << a << endl;
cout << b << endl;
while (i < n)
{
    t = b;
    b = a + b;
    cout << b << endl;
    a = t;
    i = i + 1;
}
return 0;
}

```

3.2 实验环境

本次实验基于 window10 系统下的 minGW² 环境，编译器为 GCC/G++ 编译器，编译器版本如图 1 所示（在 CMD 中执行 gcc -v 命令）。



```

C:\Users\17717>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=d:/applications/mingw/bin/./libexec/gcc/mingw32/6.3.0/
lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-6.3.0/configure --build=x86_64-pc-linux-gnu --h
ost=mingw32 --target=mingw32 --with-gmp=/mingw --with-mpfr --with-mpc=/ming
w --with-isl=/mingw --prefix=/mingw --disable-win32-registry --with-arch=i5
86 --with-tune=generic --enable-languages=c,c++,objc,obj-c++,fortran,ada --
with-pkgversion='MinGW.org GCC-6.3.0-1' --enable-static --enable-shared --e
nable-threads --with-dwarf2 --disable-sjlj-exceptions --enable-version-spec
ific-runtime-libs --with-libiconv-prefix=/mingw --with-libintl-prefix=/ming
w --enable-libstdcxx-debug --enable-libgomp --disable-libvtv --enable-nls
Thread model: win32
gcc version 6.3.0 (MinGW.org GCC-6.3.0-1)

```

图 1 gcc 版本信息

GCC 针对于从源程序到可执行文件之间的所有阶段处理均有对应的处理脚本，表 1 为一些基本的处理命令。

GCC 各阶段生成文件名后缀类型主要有 .i、.o、.s、.exe 等，.i 后缀文件为经过预处理之后的文件，.o 文件为对生成的 .s 文件进行汇编之后未链接处理的目标文件，.s 文件为汇编代码文件，.exe 为可执行程序，可在 windows 环境下直接运行。

²MinGW (Minimalist GNU for Windows)，又称 mingw32，是将 GCC 编译器和 GNU Binutils 移植到 Win32 平台下的产物，包括一系列头文件 (Win32API)、库和可执行文件。

表 1 gcc 基本编译命令

指令 op	功能
-E	激活预处理，不输出文件
-S	预处理及编译，生成汇编代码
-c	预处理、编译及汇编，生成 obj 文件
-g	调试选项
-static	静态链接，禁止使用动态库
-o	指定输出文件
-Ox(0,1,2,3)	四级优化选项

四、编译器了解初探

4.1 预处理

预处理阶段的相关实现效果见图 2 即可，该阶段主要完成对源代码的基本处理，不涉及代码语言的转换，生成的文件格式为.i 文件，该阶段的具体工作如下所示：

1. 编译器会将`#include`代码头文件进行引入，一般的 c++ 代码都会`#include<iostream>`，这些头文件常常都是非常庞大的，因此形成的预处理生成文件相较于源程序十分庞大。本次实验中使用的 `fib.cpp` 文件生成的预处理文件含有 24038 行代码。
2. 宏定义指令处理，预处理阶段会对宏定义`#define`进行处理，将所有的定义体替换为定义的内容
3. 条件编译指令处理，在预处理阶段，遇到`#ifdef`、`#elif`、`#endif`等条件编译指令时，编译器会自动进行判断处理，选择相应的代码保留。如图 2 中的宏定义 `test` 为 0，因此输出“OK!”没有执行，而是输出了“not OK!”。
4. 注释内容消除，在预处理阶段，编译器还会将所有的注释内容去除。

预处理命令为 `gcc -E fib.cpp -o fib-gcc.txt`，`g++ -E fib.cpp -o fib-gcc.txt` 值得注意的是 `gcc` 和 `g++` 在预处理阶段生成的预处理文件代码内容完全一样。

```

#include<iostream>
#define test 0
using namespace std;
int main() {
    //测试代码
    #if test
    cout<<"ok!"<<endl;
    #elif test==0
    cout<<"not ok!"<<endl;
    #endif
    cout<<"end!"<<endl;

    int a, b, i, t, n;
    a = 0;
    b = 1;
    i = 1;
    cin >> n;
    cout << a << endl;
    cout << b << endl;
    while (i < n)
    {
        t = b;
        b = a + b;
        cout << b << endl;
        a = t;
        i = i + 1;
    }
    return 0;
}

```

(a) 源代码

```

using namespace std;

int main() {

    cout<<"not ok!"<<endl;

    cout<<"end!"<<endl;
|
    int a, b, i, t, n;
    a = 0;
    b = 1;
    i = 1;
    cin >> n;
    cout << a << endl;
    cout << b << endl;
    while (i < n)
    {
        t = b;
        b = a + b;
        cout << b << endl;
        a = t;
        i = i + 1;
    }
    return 0;
}

```

(b) 预处理核心代码

图 2 预处理效果对比图

4.2 编译

编译阶段，编译器将 fib.cpp 文件或者 fib.i 文件进行编译，生成汇编代码。生成的汇编代码见附录 A 所示。该阶段执行命令为 g++(gcc) -S fib.cpp(fib.i) -o fib.s。

与预处理阶段不同的是，编译阶段编译器主要进行的工作为词法分析，语法分析，语义分析，中间代码生成，中间代码优化，目标代码生成，符号表管理，以及错误处理等。

```

#include<iostream>
#define test 0
using namespace std;
int main() {
    //测试代码
    #if test

```

```

cout>>"ok!"<<endl; //语法错误测试代码
#elif test==0
cout<<"not ok!"<<endl //词法错误
#endif
cout<<"end!"<<endl;

int a, b, i, t, n;
a = 0;
b = 1;
i = 1;
cin >> n;
cout << a << endl;
cout << b << endl;
while (i < n)
{
    t = b;
    b = a + b;
    cout << b << endl;
    a = t;
    i = i + 1;
}
return 0;
}

//上述代码直接编译报错信息如下
D:**$>g++ -S fib-test.cpp -o fib-test.s
fib-test.cpp: In function 'int main()':
fib-test.cpp:11:2: error: expected ';' before 'cout'
    cout<<"end!"<<endl;

//将词法错误的分号添加之后直接编译报错如下
D:**$>g++ -fib-test.cpp -o fib-test.s
g++: error: unrecognized command line option '-fib-test.cpp'
g++: fatal error: no input files
compilation terminated.

```

通过上述的简单修改源程序代码进行直接编译，不难看出编译器在执行编译过程

中所经历的词法分析、语法语义分析、代码优化、出错管理等操作，同时可以看出，词法分析是先于语法语义分析的，只有前者正确完成之后才会执行后者的操作。此外，在此阶段的操作中，gcc 和 g++ 一样，对程序的操作没有区别，生成的文件内容完全相同。

4.3 汇编

汇编器，主要的功能是将编译器生成的汇编程序语言翻译成为可重定位的机器码二进制语言（目标文件），该阶段执行指令为 `gcc(g++) -c fib.s -o fib.o`，生成的汇编代码如图 5 所示。

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	4c	01	07	00	00	00	00	00	f4	04	00	00	22	00	00	00	L.....?..."...□
00000010	00	00	04	01	2e	74	65	78	74	00	00	00	00	00	00	00text.....
00000020	00	00	00	00	80	01	00	00	2c	01	00	00	8c	03	00	00€....,...?..□
00000030	00	00	00	00	1f	00	00	00	20	00	30	60	2e	64	61	740`.dat
00000040	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	a.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	40	00	30	c0	2e	62	73	73	00	00	00	00	00	00	00	00	@.0?bss.....□
00000070	00	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00
00000080	00	00	00	00	00	00	00	00	80	00	30	c0	2e	72	64	61€.0?rda□
00000090	74	61	00	00	00	00	00	00	00	00	00	00	10	00	00	00	ta.....
000000a0	ac	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	?.....□
000000b0	40	00	30	40	2e	63	74	6f	72	73	00	00	00	00	00	00	@.0@.ctors.....
000000c0	00	00	00	00	04	00	00	00	bc	02	00	00	c2	04	00	00?..?..□□
000000d0	00	00	00	00	01	00	00	00	40	00	30	c0	2f	34	00	00@.0?4..□
000000e0	00	00	00	00	00	00	00	00	00	00	00	00	24	00	00	00\$....
000000f0	c0	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	?.....□
00000100	40	00	30	40	2f	31	35	00	00	00	00	00	00	00	00	00	@.0@/15.....
00000110	00	00	00	00	a8	00	00	00	e4	02	00	00	cc	04	00	00?..?..?..□e□
00000120	00	00	00	00	04	00	00	00	40	00	30	40	8d	4c	24	04@.0@.L\$.
00000130	83	e4	f0	ff	71	fc	55	89	e5	51	83	ec	34	e8	00	00	杯?q黠父Q泼4?.□H□
00000140	00	00	c7	44	24	04	01	00	00	00	00	c7	04	24	00	00	..蓼\$......?\$.□□
00000150	00	e8	00	00	00	00	c7	04	24	00	00	00	00	89	c1	e8	.?....?\$....壩?□□
00000160	00	00	00	00	83	ec	04	c7	44	24	04	09	00	00	00	c7泼.蓼\$......?□
00000170	04	24	00	00	00	00	e8	00	00	00	00	00	c7	04	24	00	\$......?\$....□□
00000180	00	00	89	c1	e8	00	00	00	00	83	ec	04	c7	45	f4	00	..壩?...泼.苔?□□H□
00000190	00	00	00	c7	45	f0	01	00	00	00	c7	45	ec	01	00	00	...苔?...苔?...□□□
000001a0	00	8d	45	e4	89	04	24	b9	00	00	00	00	e8	00	00	00	..E鉶.\$?...?..□□□
000001b0	00	83	ec	04	8b	45	f4	89	04	24	b9	00	00	00	00	e8	..泼.媯鉶.\$?...?□□
000001c0	00	00	00	00	83	ec	04	c7	04	24	00	00	00	00	89	c1泼.?\$....壩□□
000001d0	e8	00	00	00	00	83	ec	04	8b	45	f0	89	04	24	b9	00	?...泼.媯鉶.\$?□□H□
000001e0	00	00	00	e8	00	00	00	83	ec	04	c7	04	24	00	00	00	...?...泼.?\$....□□□
000001f0	00	00	89	c1	e8	00	00	00	00	83	ec	04	8b	45	e4	39	..壩?...泼.媯?□□H□
00000200	45	ec	7d	3c	8b	45	f0	89	45	e8	8b	45	f4	01	45	f0	E軛<媯鉶E鑠E?E?□□
00000210	8b	45	f0	89	04	24	b9	00	00	00	00	e8	00	00	00	00	媯鉶.\$?...?..□□□
00000220	83	ec	04	c7	04	24	00	00	00	00	89	c1	e8	00	00	00	泼.?\$....壩?...□□□
00000230	00	83	ec	04	8b	45	e8	89	45	f4	83	45	ec	01	eb	bc	..泼.媯鑠E軛E?爰□□

图 3 二进制代码（部分截图）

通过 gcc 提供的反汇编工具，可以对生成的汇编代码进行反汇编操作，直接通过汇编代码解析源代码的相关信息以及代码库引用的线索。通过命令 `gcc-nm.exe fib-g++.o` 可查看二进制代码中的名字。

```

D:**$>gcc-nm.exe fib-g++.o
00000000 b .bss
00000000 d .ctors
00000000 d .data
00000000 r .eh_frame
00000000 r .rdata
00000000 r .rdata$zzz
00000000 t .text
    U __main
00000121 t \verb|$__tcf_0$|
00000161 t __GLOBAL__sub_I_main
00000133 t __Z41__static_initialization_and_destruction_0ii
    U \verb|$__ZNSirsERi$|
    U \verb|$__ZNSolsEi$|
    U \verb|$__ZNSolsEPFRSoS_E$|
    U __ZNSt8ios_base4InitC1Ev
    U __ZNSt8ios_base4InitD1Ev
    U __ZSt3cin
    U __ZSt4cout
    U __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
00000000 r __ZStL19piecewise_construct
00000000 b __ZStL8__ioinit
    U __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
    U \verb|$_atexit$|
00000000 T _main

```

从上述反汇编得到的名字上不难发现一些特殊的函数名，如 `_main`，`__ZSt3cin`，`__ZSt4cout` 等，参照附录 A 中的汇编代码，可以看出原汇编代码中的确存在这些函数调用。

进一步通过 `objdump` 工具还可以对直接反汇编形成汇编代码，执行命令为 `objdump.exe -S fib-g++.o`，生成代码如附录 C 所示。通过对比可以看出，在反汇编的过程中，`objdump` 从 `main` 函数入口开始进行反编译，同时只对其中一些重要的基础指令以及一 I/O 指令代码进行反编译，而不是全部能够反编译成功。

4.4 链接

链接器主要是将可重定位的目标文件与一些库函数和机器二进制代码进行链接，解析二进制符号，将程序的运行首地址进行重新定位，同时将符号表中的每个符号（可重定位地址）和存储器中的一个具体位置对应。链接又分为静态链接³和动态链接⁴。

动态链接与静态链接各有优缺点。静态链接代码装载速度更快，并且不需要考虑 DLL 文件的存在与否，但是静态链接由于代码直接链接进入可执行文件，会造成可执行文件的公共代码过于庞大，造成空间浪费。动态链接基于运行时的需求由操作系统进行实时链接，能够节省内存减少页面的交换，DLL 文件和 EXE 文件彼此独立，方便代码人员进行维护和拓展，并且提高了代码的可复用性，由于这种独立性，使得开发模式更加的多样化，有利于大规模的软件开发，其缺点在于对于 DLL 模块的依赖性导致应用程序需要注重于考虑 DLL 模块和自身的版本兼容性。


 fib-g++-shared.exe	2019/9/19 12:45	应用程序	31 KB
 fib-g++-static.exe	2019/9/19 12:42	应用程序	2,089 KB

图 4 静态链接和动态链接对比

在链接阶段，gcc 和 g++ 进行代码处理形成的结果开始不同，gcc 无法将.o 文件进行链接形成可执行文件，而 g++ 能够链接形成可执行的.exe 文件，区别在 gcc 不能够自动链接 C++ 库，因此进行链接操作时候系统会进行报错。分别采用命令 g++(gcc) fib.o -o fib 执行，结果反馈见附录 B。

4.5 总结

经过预处理、编译、汇编、链接操作之后形成的中间文件如 ?? 所示，可以看到 fib-g++.i 文件占用空间较大，.exe 文件次之，.s 文件、.o 文件、.cpp 文件大小逐渐减小，这也反应了在编译器运行的不同阶段中的特点。预处理阶段编译器将庞大的头文件进行引入，造成文件变得很大，链接器将目标文件与相应的符号引用代码进行链接，最终形成的文件也会比较大一点，从汇编代码到二进制代码的汇编过程，由于机器代码本身的信息比较朴素，因此生成的二进制代码文件相较于接近人类自然语言的汇编代码大小也会更小。

³由链接器在链接时将库的内容加入到可执行程序

⁴在可执行文件装载时或运行时，由操作系统的装载程序加载库

名称	修改日期	类型	大小
fib-g++.s	2019/9/19 2:34	Assembler Source	5 KB
fib.cpp	2019/9/19 2:50	C++ Source	1 KB
fib-g++.exe	2019/9/19 3:48	应用程序	45 KB
fib-g++.i	2019/9/19 1:23	Preprocessed C/...	646 KB
fib-g++.o	2019/9/19 3:13	O 文件	3 KB

图 5 生成文件列表

五、 编译器知识进阶——调试与优化

现代编译器的发展过程中最为重要的一部分进步可能离不开程序调试这一点，具有良好调试能力的编译工具能够极大的提高编程者的编程效率。GCC/G++ 也为我们提供了很多的调试选项。表 2 列出了一些 gcc 中常用的调试选项。

表 2 gcc 基本编译命令

指令 op	功能
-g	调试选项，编译时产生调试信息
-gstabs	以 stabs 格式声称调试信息, 不包括 gdb 调试信息。
-gstabs+	以 stabs 格式声称调试信息, 包含仅供 gdb 使用的额外调试信息
-ggdb	尽可能的生成 gdb 的可以使用的调试信息

5.1 调试

在编译命令的后面加上 -g 之后生成.s 汇编文件，从图 6 可以很明显看到增加调试选项之后的生成文件变得更大了，打开文件之后可以看到类似于图 7 这样的 debug 相关语句，增加调试选项之后的文件代码行数增加了 5600 多行，多出来的代码适用于进行 gdb 调试。

fib-g++.s	2019/9/19 12:55	Assembler Source	4 KB
fib-g++-debug.s	2019/9/19 12:54	Assembler Source	90 KB

图 6 增加调试选项的编译生成文件

Ltext0:	Ldebug_info0:
.file 3 "d:/applications/mingw/lib	.long 0x35d3
.file 4 "<built-in>"	.word 0x4
.file 5 "d:/applications/mingw/lib	.secrel32 Ldebug_abbrev0
.file 6 "d:/applications/mingw/lib	.byte 0x4
.file 7 "d:/applications/mingw/lib	.uleb128 0x1
.file 8 "d:/applications/mingw/lib	.ascii "GNU C++14 6.3.0 -mtune=generic -march=i586 -g
.file 9 "d:/applications/mingw/lib	.byte 0x4
.file 10 "d:/applications/mingw/li	.ascii "fib.cpp\0"
.file 11 "d:/applications/mingw/li	.long Ltext0
.file 12 "d:/applications/mingw/li	.long Ltext0-Ltext0
.file 14 "d:/applications/mingw/li	.secrel32 Ldebug_line0
.file 15 "d:/applications/mingw/li	.uleb128 0x2
.file 16 "d:/applications/mingw/li	.ascii "std\0"
.file 17 "d:/applications/mingw/li	.byte 0x4
.file 18 "d:/applications/mingw/li	.byte 0
.file 19 "d:/applications/mingw/li	.long 0x1843
.file 20 "d:/applications/mingw/li	.uleb128 0x3
.file 21 "d:/applications/mingw/li	.ascii "__cxx11\0"
.file 22 "d:/applications/mingw/li	.byte 0x7
.file 23 "d:/applications/mingw/li	.byte 0xdf
.file 24 "d:/applications/mingw/li	.uleb128 0x4
.file 25 "d:/applications/mingw/li	.byte 0x7
.file 26 "d:/applications/mingw/ir	.byte 0xdf
.file 27 "d:/applications/mingw/ir	.long 0xa4
.file 28 "d:/applications/mingw/ir	.uleb128 0x5
.file 29 "d:/applications/mingw/ir	.byte 0x3
.file 30 "d:/applications/mingw/ir	.byte 0x40
.file 31 "d:/applications/mingw/ir	.long 0x1d4f
.file 32 "d:/applications/mingw/ir	.uleb128 0x5
.file 33 "d:/applications/mingw/ir	.byte 0x3
.file 34 "d:/applications/mingw/li	.byte 0x8b
.file 35 "d:/applications/mingw/li	.long 0x1b9a

图 7 增加 -g 选项后新增代码（部分）

5.2 优化

gcc/g++ 编译器提供了四个级别的优化水平，分别为 -O0、-O1、-O2、-O3，优化级别逐次提高，默认为 -O1 级别的优化。附录 D 中包含了 O0 和 O3 优化级别下生成的汇编代码对比。从优化代码中不难发现为了实现 O3 级别的优化，编译器添加了许多优化代码进行并行化处理，最后形成的二进制目标文件也相应增大，同时由于其因追求高的运行效率会导致一些未知的错误出现。但是在运行过程中输入 10000 甚至以上的参数求解斐波拉契数，O3 级别优化的程序相比于 O0 级别优化的程序在速度是上会有一点优势。







 fib-g++-O3.s	2019/9/19 13:36	Assembler Source	4 KB
 fib-g++-O0.exe	2019/9/19 13:35	应用程序	44 KB
 fib-g++-O0.o	2019/9/19 13:34	O 文件	2 KB
 fib-g++-O0.s	2019/9/19 13:33	Assembler Source	4 KB
 fib-g++-O3.exe	2019/9/19 13:36	应用程序	45 KB
 fib-g++-O3.o	2019/9/19 13:36	O 文件	3 KB

图 8 O3-O0 优化选项生成文件

六、 结论

通过上述的实验和分析，我们基本得到了 gcc/g++ 编译器在不同的阶段实现的功能，同时能够简单实用 gcc 的一些编译工具、反汇编工具对中间代码或二进制代码进行分析，编译器各个阶段实现的功能主要如下：有序列表是这样子的：

1. 预处理器去除注释，执行宏定义语句，同时对条件编译进行判断，还会降头文件引入与处理文件中，生成的文件代码一般较长。
2. 编译器经过词法分析、语法语义分析、中间代码生成、中间代码优化、错误处理等流程生成汇编代码
3. 汇编器将编译器生成的汇编代码转换为可重定位的二进制的目标文件
4. 链接器链接库函数和生成的目标文件，同时对目标文件中的符号引用进行处理，指定程序的开始地址。

程序的调试和优化选项主要是对编译过程进行处理，在调试选项中会生成大量的调试代码，可以通过 Gdb 工具进行调试，优化则主要是对循环递归等语句进行并行化处理，提高程序的运行速度，影响的也是编译的过程和编译形成的汇编代码，优化和调试对之后的几个阶段效率影响不大。

参考文献

- [1] 编译器的「五个十年」发展史. 云头条. [OL]. [2019-9-18].
https://mp.weixin.qq.com/s/Pg3s-AxSI84Xw_-Xt7nTA
- [2] 维基百科. 编译器. [DB/OL]. [2019-9-18]. <https://zh.wikipedia.org/wiki/編譯器>
- [3] 编译器发展及其相关介绍. [OL]. [2019-9-18].
<https://www.cnblogs.com/xkfz007/archive/2012/03/27/2420152.html>
- [4] 网络博客. 编译器的工作过程. 阮一峰. [2014-11-11].
<http://www.ruanyifeng.com/blog/2014/11/compiler.html>.
- [5] 百度百科. 动态链接. <https://baike.baidu.com/item/动态链接>.
- [6] 网络博客. GCC（警告. 优化以及调试选项）. 云水. <http://www.cnblogs.com/lsgxeva/p/7605141.html>.

附录 A 生成汇编代码 fib.s

```
.file "fib.cpp"
.section .rdata,"dr"
__ZStL19piecewise_construct:
    .space 1
.lcomm __ZStL8__ioinit,1,1
    .def __main; .scl 2; .type 32; .endef
LC0:
    .ascii "not ok!\0"
LC1:
    .ascii "end!\0"
    .text
    .globl _main
    .def _main; .scl 2; .type 32; .endef
_main:
LFB1445:
    .cfi_startproc
    leal 4(%esp), %ecx
    .cfi_def_cfa 1, 0
    andl $-16, %esp
    pushl -4(%ecx)
    pushl %ebp
    .cfi_escape 0x10,0x5,0x2,0x75,0
    movl %esp, %ebp
    pushl %ecx
    .cfi_escape 0xf,0x3,0x75,0x7c,0x6
    subl $52, %esp
    call __main
    movl $LC0, 4(%esp)
    movl $__ZSt4cout, (%esp)
    call __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
    movl $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
    movl %eax, %ecx
    call __ZNSolsEPFRSoS_E
    subl $4, %esp
    movl $LC1, 4(%esp)
```



```

movl $__ZSt4cout, (%esp)
call __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
movl $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
movl %eax, %ecx
call __ZNSolsEPFRSoS_E
subl $4, %esp
movl $0, -12(%ebp)
movl $1, -16(%ebp)
movl $1, -20(%ebp)
leal -28(%ebp), %eax
movl %eax, (%esp)
movl $__ZSt3cin, %ecx
call __ZNSirsERi
subl $4, %esp
movl -12(%ebp), %eax
movl %eax, (%esp)
movl $__ZSt4cout, %ecx
call __ZNSolsEi
subl $4, %esp
movl $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
movl %eax, %ecx
call __ZNSolsEPFRSoS_E
subl $4, %esp
movl -16(%ebp), %eax
movl %eax, (%esp)
movl $__ZSt4cout, %ecx
call __ZNSolsEi
subl $4, %esp
movl $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
movl %eax, %ecx
call __ZNSolsEPFRSoS_E
subl $4, %esp

```

L3:

```

movl -28(%ebp), %eax
cmpl %eax, -20(%ebp)
jge L2
movl -16(%ebp), %eax

```

```

movl %eax, -24(%ebp)
movl -12(%ebp), %eax
addl %eax, -16(%ebp)
movl -16(%ebp), %eax
movl %eax, (%esp)
movl $__ZSt4cout, %ecx
call __ZNSolsEi
subl $4, %esp
movl $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
movl %eax, %ecx
call __ZNSolsEPFRSoS_E
subl $4, %esp
movl -24(%ebp), %eax
movl %eax, -12(%ebp)
addl $1, -20(%ebp)
jmp L3

```

L2:

```

movl $0, %eax
movl -4(%ebp), %ecx
.cfi_def_cfa 1, 0
leave
.cfi_restore 5
leal -4(%ecx), %esp
.cfi_def_cfa 4, 4
ret
.cfi_endproc

```

LFE1445:

```

.def __tcf_0; .scl 3; .type 32; .endef

```

__tcf_0:

LFB1878:

```

.cfi_startproc
pushl %ebp
.cfi_def_cfa_offset 8
.cfi_offset 5, -8
movl %esp, %ebp
.cfi_def_cfa_register 5
subl $8, %esp

```

```

    movl $__ZStL8__ioinit, %ecx
    call __ZNSt8ios_base4InitD1Ev
    leave
    .cfi_restore 5
    .cfi_def_cfa 4, 4
    ret
    .cfi_endproc
LFE1878:
    .def __Z41__static_initialization_and_destruction_0ii; .scl 3; .type 32;
    .endef
__Z41__static_initialization_and_destruction_0ii:
LFB1877:
    .cfi_startproc
    pushl %ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    movl %esp, %ebp
    .cfi_def_cfa_register 5
    subl $24, %esp
    cmpl $1, 8(%ebp)
    jne L8
    cmpl $65535, 12(%ebp)
    jne L8
    movl $__ZStL8__ioinit, %ecx
    call __ZNSt8ios_base4InitC1Ev
    movl $__tcf_0, (%esp)
    call _atexit
L8:
    nop
    leave
    .cfi_restore 5
    .cfi_def_cfa 4, 4
    ret
    .cfi_endproc
LFE1877:
    .def __GLOBAL__sub_I_main; .scl 3; .type 32; .endef
__GLOBAL__sub_I_main:

```

LFB1879:

```
.cfi_startproc
pushl %ebp
.cfi_def_cfa_offset 8
.cfi_offset 5, -8
movl %esp, %ebp
.cfi_def_cfa_register 5
subl $24, %esp
movl $65535, 4(%esp)
movl $1, (%esp)
call __Z41__static_initialization_and_destruction_0ii
leave
.cfi_restore 5
.cfi_def_cfa 4, 4
ret
.cfi_endproc
```

LFE1879:

```
.section .ctors,"w"
.align 4
.long __GLOBAL__sub_I_main
.ident "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
.def __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc; .scl 2;
.type 32; .endef
.def __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_; .scl 2;
.type 32; .endef
.def __ZNSolsEPFRSoS_E; .scl 2; .type 32; .endef
.def __ZNSirsERi; .scl 2; .type 32; .endef
.def __ZNSolsEi; .scl 2; .type 32; .endef
.def __ZNSt8ios_base4InitD1Ev; .scl 2; .type 32; .endef
.def __ZNSt8ios_base4InitC1Ev; .scl 2; .type 32; .endef
.def _atexit; .scl 2; .type 32; .endef
```

附录 B gcc/g++ 链接 cpp 文件结果对比

```
//g++ fib.o -o fib
```

```

create file fib-g++.exe

//gcc fib.o -o fib
D:**$>gcc fib-g++.o -o fib-gcc
fib-g++.o:fib.cpp:(.text+0x21): undefined reference to `std::cout'
fib-g++.o:fib.cpp:(.text+0x26): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::operator<<
    <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&, char const*)'
fib-g++.o:fib.cpp:(.text+0x2d): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::endl<char,
    std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&)'
fib-g++.o:fib.cpp:(.text+0x34): undefined reference to
    `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
fib-g++.o:fib.cpp:(.text+0x46): undefined reference to `std::cout'
fib-g++.o:fib.cpp:(.text+0x4b): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::operator<<
    <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&, char const*)'
fib-g++.o:fib.cpp:(.text+0x52): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::endl<char,
    std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&)'
fib-g++.o:fib.cpp:(.text+0x59): undefined reference to
    `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
fib-g++.o:fib.cpp:(.text+0x7c): undefined reference to `std::cin'
fib-g++.o:fib.cpp:(.text+0x81): undefined reference to
    `std::istream::operator>>(int&)'
fib-g++.o:fib.cpp:(.text+0x8f): undefined reference to `std::cout'
fib-g++.o:fib.cpp:(.text+0x94): undefined reference to
    `std::ostream::operator<<(int)'
fib-g++.o:fib.cpp:(.text+0x9e): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::endl<char,
    std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&)'

```

```

fib-g++.o: fib.cpp: (.text+0xa5): undefined reference to
    `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
fib-g++.o: fib.cpp: (.text+0xb3): undefined reference to `std::cout'
fib-g++.o: fib.cpp: (.text+0xb8): undefined reference to
    `std::ostream::operator<<(int)'
fib-g++.o: fib.cpp: (.text+0xc2): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::endl<char,
    std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&)'
fib-g++.o: fib.cpp: (.text+0xc9): undefined reference to
    `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
fib-g++.o: fib.cpp: (.text+0xeb): undefined reference to `std::cout'
fib-g++.o: fib.cpp: (.text+0xf0): undefined reference to
    `std::ostream::operator<<(int)'
fib-g++.o: fib.cpp: (.text+0xfa): undefined reference to
    `std::basic_ostream<char, std::char_traits<char> >& std::endl<char,
    std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
    >&)'
fib-g++.o: fib.cpp: (.text+0x101): undefined reference to
    `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
fib-g++.o: fib.cpp: (.text+0x12d): undefined reference to
    `std::ios_base::Init::~Init()'
fib-g++.o: fib.cpp: (.text+0x14e): undefined reference to
    `std::ios_base::Init::Init()'
collect2.exe: error: ld returned 1 exit status

```

附录 C 反汇编生成代码

```

D:\verb\**|\verb|$| > objdump.exe -S fib-g++.o

fib-g++.o: \verb| file format pe-i386|

Disassembly of section .text:

00000000 <_main>:

```

```

0:  8d 4c 24 04      lea    0x4(%esp),%ecx
4:  83 e4 f0         and    $0xffffffff0,%esp
7:  ff 71 fc         pushl  -0x4(%ecx)
a:  55              push  %ebp
b:  89 e5           mov    %esp,%ebp
d:  51              push  %ecx
e:  83 ec 34        sub    $0x34,%esp
11: e8 00 00 00 00   call   16 <_main+0x16>
16: c7 44 24 04 01 00 00 movl  $0x1,0x4(%esp)
1d: 00
1e: c7 04 24 00 00 00 00 movl  $0x0,(%esp)
25: e8 00 00 00 00   call   2a <_main+0x2a>
2a: c7 04 24 00 00 00 00 movl  $0x0,(%esp)
31: 89 c1           mov    %eax,%ecx
33: e8 00 00 00 00   call   38 <_main+0x38>
38: 83 ec 04        sub    $0x4,%esp
3b: c7 44 24 04 09 00 00 movl  $0x9,0x4(%esp)
42: 00
43: c7 04 24 00 00 00 00 movl  $0x0,(%esp)
4a: e8 00 00 00 00   call   4f <_main+0x4f>
4f: c7 04 24 00 00 00 00 movl  $0x0,(%esp)
56: 89 c1           mov    %eax,%ecx
58: e8 00 00 00 00   call   5d <_main+0x5d>
5d: 83 ec 04        sub    $0x4,%esp
60: c7 45 f4 00 00 00 00 movl  $0x0,-0xc(%ebp)
67: c7 45 f0 01 00 00 00 movl  $0x1,-0x10(%ebp)
6e: c7 45 ec 01 00 00 00 movl  $0x1,-0x14(%ebp)
75: 8d 45 e4        lea    -0x1c(%ebp),%eax
78: 89 04 24        mov    %eax,(%esp)
7b: b9 00 00 00 00   mov    $0x0,%ecx
80: e8 00 00 00 00   call   85 <_main+0x85>
85: 83 ec 04        sub    $0x4,%esp
88: 8b 45 f4        mov    -0xc(%ebp),%eax
8b: 89 04 24        mov    %eax,(%esp)
8e: b9 00 00 00 00   mov    $0x0,%ecx
93: e8 00 00 00 00   call   98 <_main+0x98>
98: 83 ec 04        sub    $0x4,%esp

```

```

9b:  c7 04 24 00 00 00 00 movl  $0x0, (%esp)
a2:  89 c1                    mov   %eax,%ecx
a4:  e8 00 00 00 00          call  a9 <_main+0xa9>
a9:  83 ec 04                sub   $0x4,%esp
ac:  8b 45 f0                mov   -0x10(%ebp),%eax
af:  89 04 24                mov   %eax, (%esp)
b2:  b9 00 00 00 00          mov   $0x0,%ecx
b7:  e8 00 00 00 00          call  bc <_main+0xbc>
bc:  83 ec 04                sub   $0x4,%esp
bf:  c7 04 24 00 00 00 00 movl  $0x0, (%esp)
c6:  89 c1                    mov   %eax,%ecx
c8:  e8 00 00 00 00          call  cd <_main+0xcd>
cd:  83 ec 04                sub   $0x4,%esp
d0:  8b 45 e4                mov   -0x1c(%ebp),%eax
d3:  39 45 ec                cmp   %eax,-0x14(%ebp)
d6:  7d 3c                   jge   114 <_main+0x114>
d8:  8b 45 f0                mov   -0x10(%ebp),%eax
db:  89 45 e8                mov   %eax,-0x18(%ebp)
de:  8b 45 f4                mov   -0xc(%ebp),%eax
e1:  01 45 f0                add   %eax,-0x10(%ebp)
e4:  8b 45 f0                mov   -0x10(%ebp),%eax
e7:  89 04 24                mov   %eax, (%esp)
ea:  b9 00 00 00 00          mov   $0x0,%ecx
ef:  e8 00 00 00 00          call  f4 <_main+0xf4>
f4:  83 ec 04                sub   $0x4,%esp
f7:  c7 04 24 00 00 00 00 movl  $0x0, (%esp)
fe:  89 c1                    mov   %eax,%ecx
100: e8 00 00 00 00          call  105 <_main+0x105>
105: 83 ec 04                sub   $0x4,%esp
108: 8b 45 e8                mov   -0x18(%ebp),%eax
10b: 89 45 f4                mov   %eax,-0xc(%ebp)
10e: 83 45 ec 01             addl  $0x1,-0x14(%ebp)
112: eb bc                   jmp   d0 <_main+0xd0>
114: b8 00 00 00 00          mov   $0x0,%eax
119: 8b 4d fc                mov   -0x4(%ebp),%ecx
11c: c9                      leave
11d: 8d 61 fc                lea   -0x4(%ecx),%esp

```



```

120:  c3                      ret

00000121 <__tcf_0>:
121:  55                      push  %ebp
122:  89 e5                  mov   %esp,%ebp
124:  83 ec 08              sub   $0x8,%esp
127:  b9 00 00 00 00        mov   $0x0,%ecx
12c:  e8 00 00 00 00        call  131 <__tcf_0+0x10>
131:  c9                    leave
132:  c3                      ret

00000133 <__Z41__static_initialization_and_destruction_0ii>:
133:  55                      push  %ebp
134:  89 e5                  mov   %esp,%ebp
136:  83 ec 18              sub   $0x18,%esp
139:  83 7d 08 01          cmpl  $0x1,0x8(%ebp)
13d:  75 1f                jne   15e
      <__Z41__static_initialization_and_destruction_0ii+0x2b>
13f:  81 7d 0c ff ff 00 00 cmpl  $0xffff,0xc(%ebp)
146:  75 16                jne   15e
      <__Z41__static_initialization_and_destruction_0ii+0x2b>
148:  b9 00 00 00 00        mov   $0x0,%ecx
14d:  e8 00 00 00 00        call  152
      <__Z41__static_initialization_and_destruction_0ii+0x1f>
152:  c7 04 24 21 01 00 00 movl  $0x121,(%esp)
159:  e8 00 00 00 00        call  15e
      <__Z41__static_initialization_and_destruction_0ii+0x2b>
15e:  90                      nop
15f:  c9                    leave
160:  c3                      ret

00000161 <__GLOBAL__sub_I_main>:
161:  55                      push  %ebp
162:  89 e5                  mov   %esp,%ebp
164:  83 ec 18              sub   $0x18,%esp
167:  c7 44 24 04 ff ff 00 movl  $0xffff,0x4(%esp)
16e:  00

```

```

16f: c7 04 24 01 00 00 00 movl $0x1, (%esp)
176: e8 b8 ff ff ff      call 133
      <__Z41__static_initialization_and_destruction_0ii>
17b: c9                  leave
17c: c3                  ret
17d: 90                  nop
17e: 90                  nop
17f: 90                  nop

```

附录 D O3-O0 优化汇编代码对比

正在比较文件

fib-g++-O3.s 和 FIB-G++-O0.S

```

***** fib-g++-O3.s
      .file "fib.cpp"
      .section .text$__ZNKSt5ctypeIcE8do_widenEc,"x"
      .linkonce discard
      .align 2
      .p2align 4,,15
      .globl __ZNKSt5ctypeIcE8do_widenEc
      .def __ZNKSt5ctypeIcE8do_widenEc; .scl 2; .type 32; .endif
__ZNKSt5ctypeIcE8do_widenEc:
LFB1202:
      .cfi_startproc
      movzbl 4(%esp), %eax
      ret $4
      .cfi_endproc
LFE1202:
      .text
      .p2align 4,,15
      .def __tcf_0; .scl 3; .type 32; .endif
__tcf_0:
LFB1875:
      .cfi_startproc
      movl $__ZStL8__ioinit, %ecx

```

```

        jmp    __ZNSt8ios_base4InitD1Ev
        .cfi_endproc
LFE1875:
        .def    __main;        .scl    2;        .type  32;        .endif
        .section        .text.startup,"x"
        .p2align 4,,15
        .globl _main
***** FIB-G++-00.S
        .file  "fib.cpp"
        .section .rdata,"dr"
__ZStL19piecewise_construct:
        .space 1
.lcomm __ZStL8__ioinit,1,1
        .def    __main;        .scl    2;        .type  32;        .endif
        .text
        .globl _main
*****

***** fib-g++-03.s
        movl    %esp, %ebp
        pushl   %edi
        pushl   %esi
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x70,0x6
        .cfi_escape 0x10,0x7,0x2,0x75,0x7c
        .cfi_escape 0x10,0x6,0x2,0x75,0x78
        .cfi_escape 0x10,0x3,0x2,0x75,0x74
        subl    $56, %esp
        call    __main
        leal    -28(%ebp), %eax
        movl    $__ZSt3cin, %ecx
        movl    %eax, (%esp)
        call    __ZNSirsERi
***** FIB-G++-00.S
        movl    %esp, %ebp
        pushl   %ecx

```

```

.cfi_escape 0xf,0x3,0x75,0x7c,0x6
subl    $52, %esp
call    __main
movl    $0, -12(%ebp)
movl    $1, -16(%ebp)
movl    $1, -20(%ebp)
leal    -28(%ebp), %eax
movl    %eax, (%esp)
movl    $__ZSt3cin, %ecx
call    __ZNSirsERi
*****

***** fib-g++-03.s
subl    $4, %esp
movl    $__ZSt4cout, %ecx
movl    $0, (%esp)
call    __ZNSolsEi
***** FIB-G++-00.S
subl    $4, %esp
movl    -12(%ebp), %eax
movl    %eax, (%esp)
movl    $__ZSt4cout, %ecx
call    __ZNSolsEi
*****

***** fib-g++-03.s
subl    $4, %esp
movl    %eax, (%esp)
call    __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
movl    $__ZSt4cout, %ecx
movl    $1, (%esp)
call    __ZNSolsEi
***** FIB-G++-00.S
subl    $4, %esp
movl    $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
        (%esp)
movl    %eax, %ecx

```

```

    call    __ZNSolsEPFRSoS_E
    subl    $4, %esp
    movl    -16(%ebp), %eax
    movl    %eax, (%esp)
    movl    $__ZSt4cout, %ecx
    call    __ZNSolsEi
*****

***** fib-g++-03.s
    subl    $4, %esp
    movl    %eax, (%esp)
    call    __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
    cmpl    $1, -28(%ebp)
    jle     L11
    movl    $1, %esi
    movl    $1, %ebx
    xorl    %eax, %eax
    jmp     L13
    .p2align 4,,10
L17:
    movsbl  39(%eax), %ecx
L7:
    movl    %ecx, (%esp)
    movl    %edx, %ecx
    addl    $1, %esi
    call    __ZNSo3putEc
    subl    $4, %esp
    movl    %eax, %ecx
    call    __ZNSo5flushEv
    cmpl    %esi, -28(%ebp)
    movl    %ebx, %eax
    movl    %edi, %ebx
    jle     L11
L13:
    leal    (%eax,%ebx), %edi
    movl    $__ZSt4cout, %ecx
    movl    %edi, (%esp)

```

```

call    __ZNSolsEi
movl    %eax, %edx
movl    (%eax), %eax
subl    $4, %esp
movl    -12(%eax), %eax
movl    124(%edx,%eax), %eax
testl   %eax, %eax
je      L16
cmpb    $0, 28(%eax)
jne     L17
movl    %eax, %ecx
movl    %edx, -48(%ebp)
movl    %eax, -44(%ebp)
call    __ZNKSt5ctypeIcE13_M_widen_initEv
movl    -44(%ebp), %eax
movl    (%eax), %ecx
movl    24(%ecx), %edx
movl    $10, %ecx
cmpl    $__ZNKSt5ctypeIcE8do_widenEc, %edx
movl    %edx, -44(%ebp)
movl    -48(%ebp), %edx
je      L7
movl    %eax, %ecx
movl    $10, (%esp)
call    *-44(%ebp)
movl    -48(%ebp), %edx
subl    $4, %esp
movsbl  %al, %ecx
jmp     L7
.p2align 4,,10

```

L11:

```

leal    -16(%ebp), %esp
xorl    %eax, %eax
popl    %ecx
.cfi_remember_state
.cfi_restore 1
.cfi_def_cfa 1, 0

```

```

    popl    %ebx
    .cfi_restore 3
    popl    %esi
    .cfi_restore 6
    popl    %edi
    .cfi_restore 7
    popl    %ebp
    .cfi_restore 5
***** FIB-G++-00.S
    subl    $4, %esp
    movl    $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
        (%esp)
    movl    %eax, %ecx
    call    __ZNSolsEPFRSoS_E
    subl    $4, %esp
L3:
    movl    -28(%ebp), %eax
    cmpl    %eax, -20(%ebp)
    jge     L2
    movl    -16(%ebp), %eax
    movl    %eax, -24(%ebp)
    movl    -12(%ebp), %eax
    addl    %eax, -16(%ebp)
    movl    -16(%ebp), %eax
    movl    %eax, (%esp)
    movl    $__ZSt4cout, %ecx
    call    __ZNSolsEi
    subl    $4, %esp
    movl    $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
        (%esp)
    movl    %eax, %ecx
    call    __ZNSolsEPFRSoS_E
    subl    $4, %esp
    movl    -24(%ebp), %eax
    movl    %eax, -12(%ebp)
    addl    $1, -20(%ebp)
    jmp     L3

```

```

L2:
    movl    $0, %eax
    movl    -4(%ebp), %ecx
    .cfi_def_cfa 1, 0
    leave
    .cfi_restore 5
*****

***** fib-g++-03.s
    ret

L16:
    .cfi_restore_state
    call    __ZSt16__throw_bad_castv
    .cfi_endproc
***** FIB-G++-00.S
    ret
    .cfi_endproc
*****

***** fib-g++-03.s
LFE1445:
    .p2align 4,,15
    .def    __GLOBAL__sub_I_main; .scl 3;    .type 32;    .endef
__GLOBAL__sub_I_main:
LFB1876:
    .cfi_startproc
    subl    $28, %esp
    .cfi_def_cfa_offset 32
    movl    $__ZStL8__ioinit, %ecx
    call    __ZNSt8ios_base4InitC1Ev
    movl    $__tcf_0, (%esp)
    call    _atexit
    addl    $28, %esp
    .cfi_def_cfa_offset 4
    ret
***** FIB-G++-00.S
LFE1445:

```



```

        .def    __tcf_0;    .scl    3;    .type    32;    .endef
__tcf_0:
LFB1875:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $8, %esp
        movl    $__ZStL8__ioinit, %ecx
        call    __ZNSt8ios_base4InitD1Ev
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
*****

***** fib-g++-03.s
        .cfi_endproc
LFE1876:
***** FIB-G++-00.S
        .cfi_endproc
LFE1875:
        .def    __Z41__static_initialization_and_destruction_0ii; .scl 3;
        .type    32;    .endef
__Z41__static_initialization_and_destruction_0ii:
LFB1874:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $24, %esp
        cmpl    $1, 8(%ebp)
        jne     L8

```

```

    cmpl    $65535, 12(%ebp)
    jne     L8
    movl    $__ZStL8__ioinit, %ecx
    call    __ZNSt8ios_base4InitC1Ev
    movl    $__tcf_0, (%esp)
    call    _atexit
L8:
    nop
    leave
    .cfi_restore 5
    .cfi_def_cfa 4, 4
    ret
    .cfi_endproc
LFE1874:
    .def     __GLOBAL__sub_I_main; .scl 3;      .type 32;      .endef
__GLOBAL__sub_I_main:
LFB1876:
    .cfi_startproc
    pushl   %ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    movl    %esp, %ebp
    .cfi_def_cfa_register 5
    subl    $24, %esp
    movl    $65535, 4(%esp)
    movl    $1, (%esp)
    call    __Z41__static_initialization_and_destruction_0ii
    leave
    .cfi_restore 5
    .cfi_def_cfa 4, 4
    ret
    .cfi_endproc
LFE1876:
*****

***** fib-g++-03.s
    .long   __GLOBAL__sub_I_main

```

```

.lcomm __ZStL8__ioinit,1,1
    .ident "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
    .def __ZNSt8ios_base4InitD1Ev; .scl 2; .type 32; .endif
    .def __ZNSirsERi; .scl 2; .type 32; .endif
***** FIB-G++-00.S
    .long __GLOBAL__sub_I_main
    .ident "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
    .def __ZNSirsERi; .scl 2; .type 32; .endif
*****

***** fib-g++-03.s
    .def __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_;
        .scl 2; .type 32; .endif
    .def __ZNSo3putEc; .scl 2; .type 32; .endif
    .def __ZNSo5flushEv; .scl 2; .type 32; .endif
    .def __ZNKSt5ctypeIcE13_M_widen_initEv; .scl 2; .type 32;
        .endif
    .def __ZSt16__throw_bad_castv; .scl 2; .type 32; .endif
    .def __ZNSt8ios_base4InitC1Ev; .scl 2; .type 32; .endif
***** FIB-G++-00.S
    .def __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_;
        .scl 2; .type 32; .endif
    .def __ZNSolsEPFRSoS_E; .scl 2; .type 32; .endif
    .def __ZNSt8ios_base4InitD1Ev; .scl 2; .type 32; .endif
    .def __ZNSt8ios_base4InitC1Ev; .scl 2; .type 32; .endif
*****

```