编译C程序的工作原理

February 17, 2017

C是一种高级语言，它需要编译器将其转换为可执行代码，以使得程序能在机器上运行。

1 如何编译和运行一个C程序？

以下介绍在MAC或Linux上使用gcc编译器的几个步骤。

- 首先使用编辑器（如vi或emacs等）创建一个C程序，并将其保存为add2num.c。

```
$ emacs add2num.c
```

在编辑界面输入以下内容：

```
// program for the addition of two numbers
#include<stdio.h>
#define add(a,b) (a+b) //using macros
int main(void)
{
  int a = 5, b = 4;
  printf("Addition is: %d\n", add(a,b));
  return 0;
}
```

- 然后用以下命令编译，并查看当前目录下的文件。

```
$ gcc -Wall add2num.c o add2num
$ ls
add2num     add2num.c
```

选项-Wall启动所有编译器的警告信息。建议使用该选项以生成更好的代码。 选项-o用来制定输出文件名。如果缺省该选项，则输出文件将默认为a.out。

- 编译通过后，将会生成可执行文件，可使用以下命令来运行生成的可执行文件。

```
$ ./add2num
```

2 编译过程中发生了什么？

编译器将一个C程序转换为一个可执行文件，这经历了4个阶段：

- 预处理
- 编译
- 汇编
- 链接

执行以下命令，在当前目录下会生成所有的中间文件以及可执行文件。

```
$gcc -Wall -save-temps filename.c o filename
$ls
add2num      add2num.c    add2num.o
add2num.bc   add2num.i    add2num.s
```

接下来，让我们一个个地来看这些中间文件中的内容。

- 预处理

  这是源代码后的第一阶段，包括

  - 去掉注释
  - 宏的展开
  - 头文件的展开

  预处理的输出保存在文件add2num.i中，可用以下命令来查看其内容：

```
$less add2num.i
$ls
...
int printf(const char * restrict, ...) __attribute__((__format__ (__printf__,
1, 2)));
...
extern int __vsnprintf_chk (char * restrict, size_t, int, size_t,
       const char * restrict, va_list);
# 499 "/usr/include/stdio.h" 2 3 4
# 3 "add2num.c" 2

int main(void)
{
  int a = 5, b = 4;
  printf("Addition is: %d\n", (a+b));
  return 0;
}
```

  分析：在以上内容中，源文件被附加了很多信息，但在末尾代码仍被保留。printf()函数中包含了a+b而非add(a,b)，因为宏已被展开。注释被去除掉。#include<stdio.h>没有了，取而代之的是很多代码。因此，头文件已被展开，并且被包含到了源文件中。

- 编译

  接下来就是编译add2num.i，并生成一个编译过的输出文件add2num.s。该文件为汇编指令，可用以下命令查看：

```
$less add2num.s
...
Ltmp2:
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        leaq    L_.str(%rip), %rdi
        movl    $0, -4(%rbp)
        movl    $5, -8(%rbp)
        movl    $4, -12(%rbp)
        movl    -8(%rbp), %eax
        addl    -12(%rbp), %eax
        movl    %eax, %esi
        movb    $0, %al
        callq   _printf
        xorl    %esi, %esi
...
```

以上内容表明这是汇编语言，能为汇编器所识别。

- 汇编

  这一阶段，将通过汇编器将filename.s转换成filename.o。该文件包含机器指令。需要注意的是，该阶段只会将现有代码转换成机器语言，而诸如printf()d的函数调用则不会。

```
$less add2num.o
<CF><FA><ED><FE>^G^@^@^A^C^@^@^@^A^@^@^@^D^@^@^@^@^B^@^@^@ ^@^@^@^@^@^@^Y^@^@^
@<88>^A^@^@^@^@^@^@^@^
...
```

- 链接

  这是最后一个阶段，将完成所有函数调用及其定义的链接工作。链接器知道所有这些函数在何处执行。链接器也会做一些额外的工作，以添加一些启动和结束程序所需的额外代码。 在命令行中输入以下命令，可看出从目标文件到可执行文件时文件大小的变化。这是因为链接器为我们的程序添加了额外的代码。

```
$ size add2num.o
__TEXT  __DATA  __OBJC  others  dec hex
145 0   0   32  177 b1
HXM:Code hxm$ size add2num
__TEXT  __DATA  __OBJC  others  dec hex
4096    4096    0   4294971392  4294979584  100003000
```