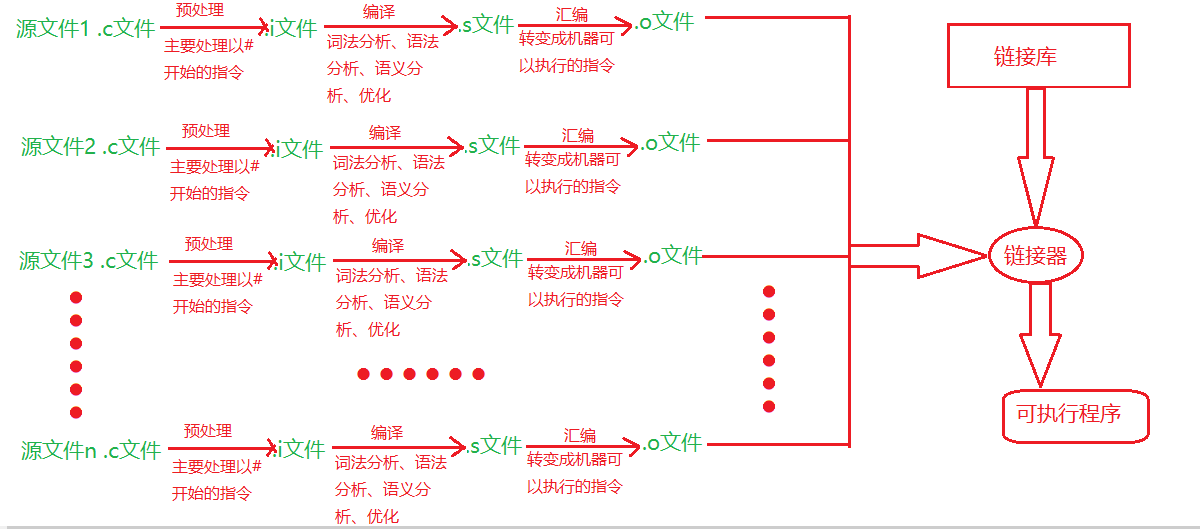
对程序编译的一点认识

认识程序的编译执行过程是学习编程的开端！只有能够深入了解程序的运行机制，了解编译过程，清楚在这个过程中编译器做了什么事，才能在程序出现错误时游刃有余的解决。

   总的来说，程序的运行包括四个过程：预处理、编译、汇编和链接。



在这，我用liunx中使用gcc来编译“hello world”程序做一演示，理解编译的原理。

在liunx的终端中使用vim编辑器，建立一个hello.c源程序

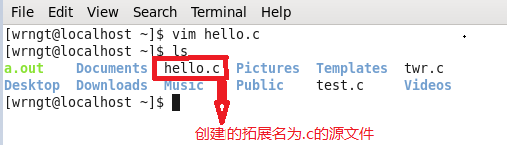
使用命令：$ vim hello.c

此时进入vim编辑器，在vim中我们写好源代码：

**[cpp]** [view plain](http://blog.csdn.net/WRNGT/article/details/54766824) [copy](http://blog.csdn.net/WRNGT/article/details/54766824)

1. #include<stdio.h>
2. **int** main()
3. {
4. printf("hello world\n");
5. **return** 0;
6. }

在vim编辑器中使用wq保存并退出，此时，我们可以用命令ls显示找到编辑的源文件：



接下来就开始进入程序运行的过程了：

1.预编译阶段

使用命令：

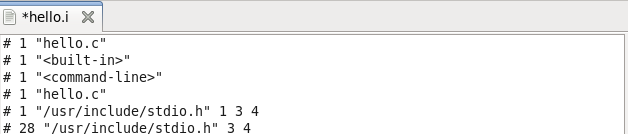
**[cpp]** [view plain](http://blog.csdn.net/WRNGT/article/details/54766824) [copy](http://blog.csdn.net/WRNGT/article/details/54766824)

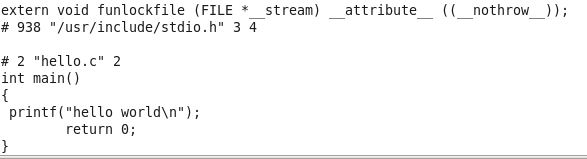
1. $ gcc -E hello.c -o hello.i

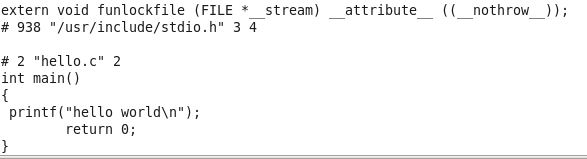
在此过程中，源文件拓展名为.c，而进行预编译之后的拓展名为.i，形成一个.i文件，



我们可以使用命令pwd找到这个预编译文件的目录，并且打开这个文件。







经过分析此文件内容，我们可以清楚看到，在预编译阶段主要进行的工作：

* 将所有的#define删除，并且展开所有的宏定义
* 处理所有的条件预编译指令，比如#if #ifdef #elif #else #endif等
* 处理#include 预编译指令，将被包含的文件插入到该预编译指令的位置。
* 删除所有注释 “//”和”/\* \*/”.
* 添加行号和文件标识，以便编译时产生调试用的行号及编译错误警告行号。
* 保留所有的#pragma编译器指令，因为编译器需要使用它们

经过预编译后的.i文件不含任何宏定义，所有的宏已经被展开并且插入到.i文件中。

2.编译阶段

使用命令:

**[cpp]** [view plain](http://blog.csdn.net/WRNGT/article/details/54766824) [copy](http://blog.csdn.net/WRNGT/article/details/54766824)

1. $ gcc -S hello.i -o hello.s

最终得到一个拓展名为.s的文件



同样，我们打开此文件：



在此过程经过一系列的词法分析、语法分析、语义分析及优化最终得到一个汇编代码文件。

注：词法分析：源代码进入扫描器，进而扫描器将代码的字符分割成一系列的记号（关键字、标识符、字面量、特殊符号如加号减号），同时，标识符放符号表，数字、字符串放文字表。

       语法分析：对扫描器产生的记号进行语法分析，产生语法数，生成以表达式为节点的树，同时，很多运算符号的优先级和含义也被确定，以及区分多重含义符号，若出现表达式不合法(括号不匹配、缺少操作符等),编译器将报告语法分析阶段的错误。

        语义分析：此时一般对声明和类型的匹配、类型之间的转换进行分析，类型不匹配，就会报错；以及分析与语义相关的问题，比如：0作为除数等，错误进行报错。

3.汇编阶段：此阶段相对编译器比较简单，它没有复杂的语法、语义以及优化，只是根据汇编指令和机器指令的对照表一一翻译即可。

使用命令：

**[cpp]** [view plain](http://blog.csdn.net/WRNGT/article/details/54766824) [copy](http://blog.csdn.net/WRNGT/article/details/54766824)

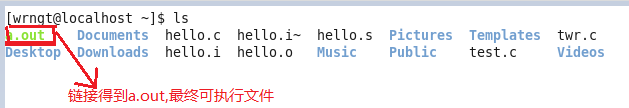
1. $ gcc -c hello.s -o hello.o



在此过程中将汇编代码的.s文件经过汇编器转变成机器可以执行的指令，每一个汇编语句几乎都对应一条机器指令。

4.链接阶段：

**链接器**(linker)将一个个的目标文件(或许还会有若干程序库)链接在一起生成一个完整的可执行文件a.out



使用命令：

**[html]** [view plain](http://blog.csdn.net/WRNGT/article/details/54766824) [copy](http://blog.csdn.net/WRNGT/article/details/54766824)

1. $ ./a.out

即可运行此程序：



总结：

       在这整个过程中编译过程可分为6步：扫描（词法分析）、语法分析、语义分析、源代码优化、代码生成、目标代码优化。

       链接的主要内容是把各个模块之间相互引用的部分处理好，使得各个模块之间能够正确地衔接。

       链接的主要过程包括：地址和空间分配，符号决议，重定位等。

       同时，链接又分为静态链接和动态链接。静态链接是指在编译阶段直接把静态库加入到可执行文件中去，这样可执行文件会比较大。而动态链接则是指链接阶段仅仅只加入一些描述信息，而程序执行时再从系统中把相应动态库加载到内存中去。