一、 C 语言编译过程

C 语言的编译过程可分为四个阶段：

1、预处理（Preprocessing）

对源程序中的**伪指令**（即以#开头的指令）和**特殊符号**进行处理的过程。

伪指令包括：1）宏定义指令；

2）条件编译指令；

3）头文件包含指令；

2、编译（Compilation）

编译就是将**源程序**转换为计算机可以执行的**二进制代码**。

说明：

在 Linux 下，**目标文件**的缺省后缀为.o

编译程序将通过词法分析和语法分析，将其翻译成为等价的汇编代码。

在使用 gcc 进行编译时，缺省情况下，不输出这个汇编代码的文件。如果需要，可以在编译时指定-S 选项。这样，就会输出同名的汇编语言文件。

3、汇编（Assembly）

汇编的过程实际上是将**汇编语言代码**翻译成**机器语言**的过程。

产生一个扩展名为.o 的目标文件。

4、链接（Linking）

目标代码不能直接执行，要想将目标代码变成可执行程序，还需要进行链接操作。才会生成真正可以执行的可执行程序。

链接操作最重要的步骤就是将函数库中相应的代码组合到目标文件中。

二、文件名后缀

gcc 可以针对支持不同的源程序文件进行不同的处理，文件格式以文件的后缀来识别。

文件名后缀	文件类型
.c	C 源文件
.C .cpp .cc .c++ .cxx	C++源文件
.h	头文件
.i	预处理后的 C 源文件
.s	汇编程序文件
.o	**目标文件**
.a	静态链接库

.so	动态链接库

三、 gcc 编译器简介

gcc（GNU Compiler Collection）

在 Linux 平台上最常用的 C 语言编译系统是 gcc，它是 GNU 项目中符合 ANSI C 标准的编译系统。

gcc 的使用格式：

```
gcc [options][filenames]
```

说明：当不用任何选项时，gcc 将会生成一个名为 a.out 的可执行文件。

例子：在 linux 上编译一个 c 程序（文件名为 hello.c ；执行 gcc hello.c）。

```
#include
int main()
{
        printf("hello world.\n");
            return 0;
}
```

运行编译好的可执行 c 文件命令是./a.out

四、gcc 编译器的工作过程

1、预处理（Preprocessing）

2、编译（Compilation & Assembly）

源代码转换**为汇编语言**（在编译时选择-S 选项，可以看到生成的汇编代码.s 文件）

汇编代码（.s）转换为**目标代码**（.o）

3、链接（Linking）

将**目标代码**与各**库函数**进行链接并重定位，生成可执行程序。

五、gcc 命令行选项

1、预处理选项

选项	说明
-D name	定义一个宏 name，并可以指定值
-I dir	指定头文件的路径 dir。先在指定的路径中搜索要包含的头文件，若找不到，则在标准路径（/usr/include，/usr/lib 及当前工作目录）上搜索。
-E	只对文件进行预处理，不进行编译、汇编、链接，生成的结果送标准输出 即：只运行 C 预编译器

-o file	将输出写到指定的文件 file 中 即：产生目标（.i 、.s 、 .o 、可执行文件等）

例子：使用 -I 选项包含保存在非标准位置中的头文件。

```
# gcc -I/usr/openwin/include file.c
```

例子：使用-D 选项定义宏，其作用等价于在源文件中使用宏定义指令。

```
main()
{
                printf("display -D variable %s\n",DOPTION);
                printf("hello,everybody!!\n");
}
# gcc -D DOPTION='"testing -D"' hello.c
```

2、编译程序选项

选项	说明
-o file1 file2	将文件 **file2** 编译成可执行文件 **file1**。 如果未使用该选项，则可执行文件放在 **a.out** 中
-S	只进行编译，不进行汇编，生成汇编代码文件扩展名为.s 即：告诉编译器产生汇编语言文件后停止编译
-c	只把源文件编译成目标代码.o，不进行汇编、链接。 用于实现对源文件的分别编译
-g	在目标代码中加入供调试程序 gdb 使用的附加信息
-v	显示 gcc 版本
-Wall	显示警告信息

例子：在 gcc 中使用-W 控制警告信息。

```
# gcc -Wall -o hello1 hello1.c
```

例子：使用 gcc 的-g 选项来产生调试符号，

```
# gcc -g -o test1 test1.c
```

例子：多文件的编译。

```
//meng1.c
#include
main()
{
                int r;
                printf("enter an integer,please!\n");
                scanf("%d",&r);
```

```
                square(r);
                return 0;
}
//meng2.c
#include
int square(int x)
{
                printf("The square=%d\n",x*x);
                return (x*x);
}
```

编译方法一：

```
# gcc -c meng1.c
# gcc -c meng2.c
# gcc meng1.o meng2.o -o meng12
```

编译方法二：

```
# gcc -o meng13 meng1.c meng2.c
```

说明：

方法二不产生中间目标文件，直接生成一个可执行文件，因而，程序内容稍有改动，就要重新编译全部程序。

3、优化程序选项

优化是编译器的一部分，它可以检查和组合编译器生成的代码，指出未达到最优的部分，并重新生成它们，从而使用户编写的程序更加完美且节省空间。

在 gcc 编译器选项中，使用-O 选项对代码进行优化。

优化级别分 3 级，由高到低分别为：-O3、-O2、-O1，

优化程序选项

选项	说明
-O1(-O)	对编译出的代码进行优化
-O2	进行比-O 高一级的优化
-O3	产生更高级别的优化

说明：

-O1（或-O）、-O2、-O3 分别代表优化级别，数字越高，代表 gcc 的优化级别越高，高的优化级别代表着程序将运行的更快。

优化级别越高则程序量越大。

直接优化程序本身，性能的提高的变化更加明显。

4、连接程序选项

库：是一组预先编译好的函数集合。

说明：

标准库文件一般存储在/lib 和/usr/lib 目录中。

所有的库名都以 lib 开头。例如：libc.so（标准 C 语言函数库）、libm.so（数学运算函数库）

以.a 结尾的是静态库；以.so 结尾的库是动态库。

使用 ar 工具将目标文件收集起来，放到一个归档文件中。

连接程序选项

选项	说明
-L dir	将 dir 所指出的目录加到"函数库搜索列表"中
-llib	链接 lib 库
-I name	连接时，加载名字为 name 的函数库。该库位于系统预设的目录或者由-L 选项确定的目录下。 实际的库名是 libname（后缀为.a 或.so）

说明：

链接过程通常的形式如下：

gcc -o file file.o -L dirname -lxxx

-L：指定了链接时用到的库文件所在的目录。

-lxxx：指示链接的库函数名为 libxxx.a

例子：编译产生可执行文件 hello，搜索数学库以解决问题。

 # gcc -o hello hello.c /usr/lib/libm.a

或者

 # gcc -o hello hello.c -lm

例子：创建一个小型库

包含两个函数 pro1、pro2，然后在示例程序中调用其中一个函数。

```
#include
void pro1(int arg)
{
printf("hello:%d\n", arg);
}
#include
void pro2(char *arg)
{
printf("welcome to:%s", arg);
}
void pro1(int);
```

```
void pro2(char *);
#include "lib.h"
int main()
{
pro2("Linux world.");
exit(0);
}
```