可以把隐含规则中使用的变量分成两种，一种是命令相关的，如”CC”,一种是参数相关，如”CFLAGS”，下面是所有隐含规则中用到的变量：

AR 函数库打包程序（ar）

AS 汇编语言编译程序（as）

CC c语言编译程序（cc）

CXX C++语言编译程序（g++）

CO 从RCS文件中扩展文件程序（co）

CPP C程序的预处理器（标准输出设备）。（$(CC) -E）

FC Fortran和Ratfor的编译器和预处理程序。（f77）

GET 从SCCS文件中扩展文件的程序（get）

LEX Lex方法分析器程序（针对c或Ratfor），（lex）

PC Pascal语言编译程序。（pc）

YACC Yacc文件分析器（针对c程序）。（yacc）

YACCR Yacc文件分析器（针对Ratfor）（yacc -r）

MAKEINFO 转换Texinfo源文件（.texi）到Info文件程序（makeinfo）

TEX 从Tex源文件创建Tex DVI文件的程序（tex）

TEXI2DVI 从Texinfo源文件创建Tex DVI文件的程序（texi2dvi）

WEAVE 转换Web到TeX的程序。（weare）

**库操作选项**

1．-I选项

可以向gcc的头文件搜索路径中添加新的目录。Eg，如果在/home/dong/include/目录下有编译时所用到的头文件，为了让gcc能够顺利的找到他们。就可以使用-I 选项

gcc foo.c –I /home/dong/include/ -o foo

2.-L选项

如果使用了不在标准库的头文件，那么可以通过-L选项向gcc的库文件搜索路径中添加新的目录，eg：如果在/home/dong/lib/目录下有链接时所需要的库文件lifoo.so，为了让gcc能够顺利地找到它，可以使用下面的命令：

gcc foo.c –L /home/dong/lib -foo -o foo

3.-static选项

Linux下的库文件分为静态库（.a）和动态库（.so）。gcc在链接的时候会优先使用动态链接库。加上-static表示强制使用静态链接库

gcc foo.c –L /home/dong/lib –static -lfoo -o foo

4．-shared选项

生成一个共享的文件目标，它能够与其他的目标一起生成链接生成一个可执行的文件

（1）。生成目标文件

gcc -Wall –c –fpic file1.c file2.c file3.c

-fpic：指定生成的.o目标文件可被重定址（pic position independent code）

（2）：生成动态库文件（-Wall 能够使gcc产生尽可能多的警告信息）

gcc –shared –o libNAME.so file1.c file2.c file3.c

上述的两条命令可以合并为：

gcc –Wall –shared -fpic –o libNAME.so file1.c file2.c file3.c

-O 对程序进行优化编译，链接，使得可执行文件的执行效率提高，但编译，链接的速度相应的慢了下来.

gcc test.c –lm –o test –lm表示连接系统的数学库。

GCC/G++编译选项

|  |  |
| --- | --- |
| -c | 只进行预处理、编译和汇编，生成.o文件 |
| -S | 只进行预处理和编译，生成.s文件 |
| -o | 指定目标名称，常与-c –S同时使用，默认是.out |
| -Idir | 优先在选项后的目录中查找包含的头文件 |
| -lname | 链接后缀为.so(或.a)的链接库赖编译程序 |
| -O[0-3] | 编译器优化 |
| -shared | 使用动态库 |
| -static | 禁止使用动态库 |

**预处理 编译 汇编 链接 执行**

Linux 文件状态和属性操作

struct stat结构体是stat.h函数库提供的一种用于保存文件类型的结构体。结构成员如下：

|  |  |
| --- | --- |
| st\_mode | 文件权限与文件类型信息 |
| st\_ino | 与该文件关联的inode |
| st\_dev | 文件保存在其上的设备 |
| st\_dev | 文件所有者的用户身份标识 |
| st\_uid | 文件群组的分组身份标识 |
| st\_atime | 上次被访问时间 |
| st\_ctime | 文件权限、所有者、群组或内容上面上次被修改时间 |
| st\_mtime | 文件内容方面上次被修改时间 |
| st\_nlink | 该文件上硬链接的个数 |

st\_mode 与其他成员相比要复杂的多，必须使用标志与之进行按位运算才能获得相应信息。

|  |  |
| --- | --- |
| S\_IFBLK | 特殊的块设备 |
| S\_IFDIR | 目录 |
| S\_IFCHR | 特殊的字符设备 |
| S\_IFIFO | FIFO设备（管道） |
| S\_IFREG | 普通文件 |
| S\_IFLINK | 符号连接 |
| S\_ISUID | 设置了SUID位 |
| S\_ISGID | 设置了SGID位 |
| S\_ISVTX | 设置了sticky位 |
| S\_IFMT | 文件类型 |
| S\_IRWXU | 属主的读/写/执行权限。可以分为S\_IXUSR,S\_IRUSR,S\_IWUSR |
| S\_IRWXG | 属主的读/写/执行权限。可以分为S\_IXGRP,S\_IRGRP,S\_IWGRP |
| S\_IRWXO |  |

S\_ISLINK(st\_mode) 判断是否位符号连接

S\_ISREG(st\_mode) 是否为一般文件

S\_ISDIR(st\_mode) 是否为目录

S\_ISCHR(st\_mode) 是否为字符装置文件

S\_ISBLK(s3e) 是否为先进先出

S\_ISSOCK(st\_mode) 是否位socket

获取文件状态和属性操作可使用下面3个函数。

fstat(文件标识符,struct stat\*buf) //返回一个已打开文件的状态和属性信息

/\*对未打开文件操作，lstat和stat的区别是当文件是一个符号链接时，lstat返回的是符号链接本身的信息，而stat返回的是该符号链接指向的文件的信息。\*/

lstat(文件标识符,struct stat\*buf)

stat(文件标识符,struct stat\*buf)