# Makefile学习笔记

## Makefile

GNU make 地址

参考文档：https://www.gnu.org/software/make/manual/make.html

ifneq ($(EXE), "")

ifneq与($(EXE), "")之间要有空格

## gcc从源码到可执行文件的步骤

大体分为一下四个步骤

1. 第一步、预处理阶段：

处理源码中的预处理语句（总是以#打头，如#include、#if #elif #end #error、#pragma pack|comment等）；

gcc -E test.c -o test.i 或 gcc -E test.c

1. 第二步、编译阶段：

通过词法分析和语法分析，确认所有指令都符合语法规则后，将其转换成功能等效的汇编语言代码；

gcc -S test.i -o test.s

1. 第三步、汇编阶段：

使用汇编器对汇编代码进行处理，生成机器语言代码，保存在后缀为.o的目标文件中。

gcc -c test.s -o test.o

1. 第四步、链接阶段：

将各个模块之间相互作用的部分都处理好，使得各个模块之间能够正确地衔接。

gcc test.o -o test

在Linux下进行C语言编程，必然要采用GNU GCC来编译C源代码生成可执行程序。

一、GCC快速入门

Gcc指令的一般格式为：Gcc [选项] 要编译的文件 [选项] [目标文件]

其中，目标文件可缺省，Gcc默认生成可执行的文件名为：编译文件.out

我们来看一下经典入门程序"Hello World！"

# vi hello.c

#include <stdlib.h>

#include <stdio.h>

void main(void)

{

printf("hello world!\r\n");

}

用gcc编译成执行程序。

#gcc -o hello hello.c

该命令将hello.c直接生成最终二进制可执行程序a.out

这条命令隐含执行了（1）预处理、（2）汇编、(3)编译并(4)链接形成最终的二进制可执行程序。这里未指定输出文件，默认输出为a.out。

如何要指定最终二进制可执行程序名，那么用-o选项来指定名称。比如需要生成执行程序hello.exe

那么

#gcc hello.c -o hello.exe

二、GCC的命令剖析--四步走

从上面我们知道GCC编译源代码生成最终可执行的二进制程序，GCC后台隐含执行了四个阶段步骤。

GCC编译C源码有四个步骤：预处理-----> 编译 ----> 汇编 ----> 链接

现在我们就用GCC的命令选项来逐个剖析GCC过程。

1）预处理(Pre-processing)

在该阶段，编译器将C源代码中的包含的头文件如stdio.h编译进来，用户可以使用gcc的选项”-E”进行查看。

用法:#gcc -E hello.c -o hello.i

作用：将hello.c预处理输出hello.i文件。

[root]# gcc -E hello.c -o hello.i

[root]# ls

hello.c hello.i

[root]# vi hello.i

# 1 "hello.c"

# 1 "<built-in>"

# 1 "<command line>"

# 1 "hello.c"

# 1 "/usr/include/stdlib.h" 1 3

# 25 "/usr/include/stdlib.h" 3

# 1 "/usr/include/features.h" 1 3

# 291 "/usr/include/features.h" 3

# 1 "/usr/include/sys/cdefs.h" 1 3

# 292 "/usr/include/features.h" 2 3

# 314 "/usr/include/features.h" 3

# 1 "/usr/include/gnu/stubs.h" 1 3

# 315 "/usr/include/features.h" 2 3

# 26 "/usr/include/stdlib.h" 2 3

# 3 "hello.c" 2

void main(void)

{

printf("hello world!\r\n");

}

2)编译阶段(Compiling)

第二步进行的是编译阶段，在这个阶段中，Gcc首先要检查代码的规范性、是否有语法错误等，以确定代码的实际要做的工作，在检查无误后，Gcc把代码翻译成汇编语言。用户可以使用”-S”选项来进行查看，该选项只进行编译而不进行汇编，生成汇编代码。

选项 -S

用法：[root]# gcc –S hello.i –o hello.s

作用：将预处理输出文件hello.i汇编成hello.s文件。

[root@richard hello-gcc]# ls

hello.c hello.i hello.s

如下为hello.s汇编代码

[root@richard hello-gcc]# vi hello.s

.file   "hello.c"

.section    .rodata

.LC0:

.string "hello world!\r\n"

.text

.globl main

.type   main,@function

main:

pushl   %ebp

movl    %esp, %ebp

subl    $8, %esp

andl    $-16, %esp

movl    $0, %eax

subl    %eax, %esp

subl    $12, %esp

pushl   $.LC0

call    printf

addl    $16, %esp

movl    $0, %eax

leave

ret

.Lfe1:

.size   main,.Lfe1-main

.ident "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"

3)汇编阶段(Assembling)

汇编阶段是把编译阶段生成的”.s”文件转成二进制目标代码.

选项 -c

用法：[root]# gcc –c hello.s –o hello.o

作用：将汇编输出文件test.s编译输出test.o文件。

[root]# gcc -c hello.s -o hello.o

[root]# ls

hello.c hello.i hello.o hello.s

4）链接阶段(Link)

在成功编译之后，就进入了链接阶段。

无选项链接

用法：[root]# gcc hello.o –o hello.exe

作用：将编译输出文件hello.o链接成最终可执行文件hello.exe。

[root]# ls

hello.c hello.exe hello.i hello.o hello.s

运行该可执行文件，出现正确的结果如下。

[root@localhost Gcc]# ./hello

Hello World!

在这里涉及到一个重要的概念：函数库。

读者可以重新查看这个小程序，在这个程序中并没有定义”printf”的函数实现，且在预编译中包含进的”stdio.h”中也只有该函数的声明，而没有定义函数的实现，那么，是在哪里实现”printf”函数的呢？最后的答案是：系统把这些函数实现都被做到名为libc.so.6的库文件中去了，在没有特别指定时，gcc会到系统默认的搜索路径”/usr/lib”下进行查找，也就是链接到libc.so.6库函数中去，这样就能实现函数”printf” 了，而这也就是链接的作用。

你可以用ldd命令查看动态库加载情况：

[root]# ldd hello.exe

libc.so.6 => /lib/tls/libc.so.6 (0x42000000)

/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

函数库一般分为静态库和动态库两种。静态库是指编译链接时，把库文件的代码全部加入到可执行文件中，因此生成的文件比较大，但在运行时也就不再需要库文件了。其后缀名一般为”.a”。动态库与之相反，在编译链接时并没有把库文件的代码加入到可执行文件中，而是在程序执行时由运行时链接文件加载库，这样可以节省系统的开销。动态库一般后缀名为”.so”，如前面所述的libc.so.6就是动态库。gcc在编译时默认使用动态库。

## 参考资料

版权声明：本文参考了其他CSDN博主的文章，遵循CC 4.0 BY-SA版权协议，现附上原文出处链接及本声明。

1. Makefile入门一、helloworld 原文链接：https://blog.csdn.net/hjxu2016/article/details/100672126
2. https://www.cnblogs.com/qytan36/archive/2010/05/25/1743955.html