

gcc/g++：编译器

功能

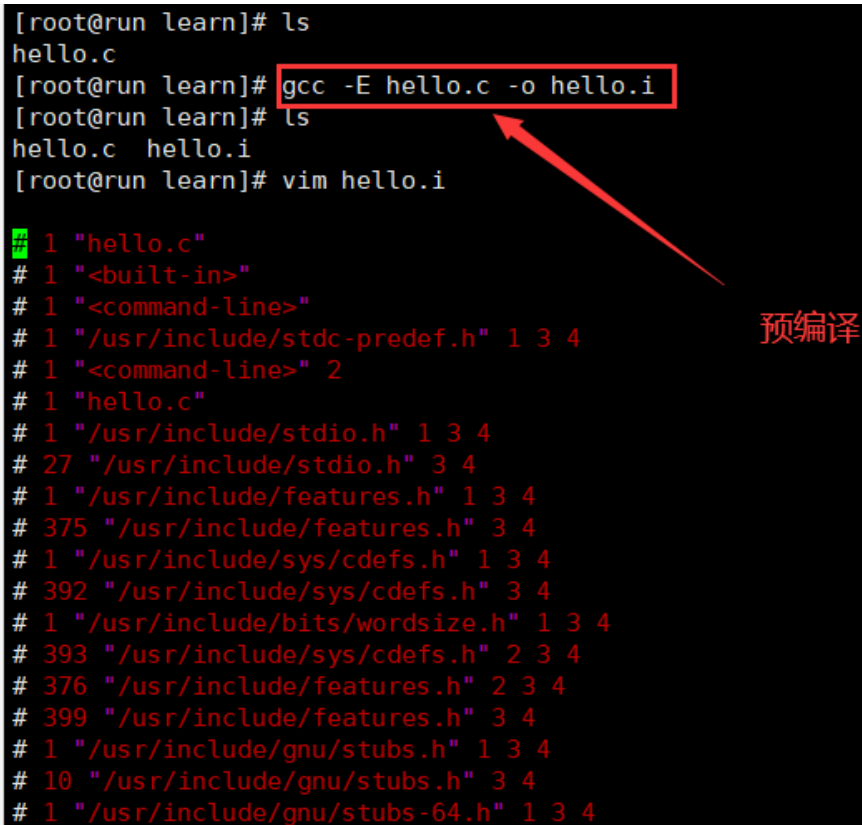
将C/C++高级语言代码翻译成机器可识别的代码

编译过程

1. 预处理(进行宏替换)

宏定义展开，头文件展开、条件编译等，同时将代码中的注释删除

eg: gcc -E hello.c -o hello.i



```
[root@run learn]# ls
hello.c
[root@run learn]# gcc -E hello.c -o hello.i
[root@run learn]# ls
hello.c  hello.i
[root@run learn]# vim hello.i

1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 375 "/usr/include/features.h" 3 4
# 1 "/usr/include/sys/cdefs.h" 1 3 4
# 392 "/usr/include/sys/cdefs.h" 3 4
# 1 "/usr/include/bits/wordsize.h" 1 3 4
# 393 "/usr/include/sys/cdefs.h" 2 3 4
# 376 "/usr/include/features.h" 2 3 4
# 399 "/usr/include/features.h" 3 4
# 1 "/usr/include/gnu/stubs.h" 1 3 4
# 10 "/usr/include/gnu/stubs.h" 3 4
# 1 "/usr/include/gnu/stubs-64.h" 1 3 4
```

2. 编译(生成汇编)

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，把代码翻译成汇编语言

eg: gcc -S hello.i -o hello.s

库文件：已经汇编完成的公共代码

链接：将库中的代码拿到要生成的可执行程序中

静态链接：链接静态库，将库中的代码直接全部拷贝到可执行程序中；好处：程序运行时，不需要依赖库文件的存在；缺点：占用资源多

动态链接：链接动态库，只是记录库中的接口符号位置信息；好处：资源占用冗余较小；缺点：程序运行时需要动态库存在。

一步编译：gcc hello.c -o hello

gdb：调试器

功能

调试一个程序的运行过程

使用

Linux下程序编译默认生成release版本程序，不带有调试符号信息，意味着程序无法被调试；所以程序调试的前提就是编译生成debug版本程序，因此gcc编译程序时需要加上 -g 选项，开启调试，向程序中添加调试符号信息。

1. gcc编译程序时需要加上 -g 选项，向程序中添加调试符号信息

gcc -g hello.c -o hello

```
[root@run learn]# ls
hello.c
[root@run learn]# gcc -g hello.c -o hello
[root@run learn]# ls
hello hello.c
[root@run learn]#
```

2. gdb调试程序，需要将程序信息，加载到gdb中，进入gdb

gdb ./hello

```
[root@run learn]# ls
hello hello.c
[root@run learn]# gdb ./hello
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-119.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/learn/hello...done.
(gdb)
```

进入gdb，并将程序信息加载到gdb中

或者 gdb 进入后 file ./hello

```
[root@run learn]# ls
hello hello.c
[root@run learn]# gdb
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-119.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
(gdb) file ./hello
Reading symbols from /root/learn/hello...done.
(gdb)
```

先进入gdb

将需要的程序信息，加载到gdb中

3. 开始调试

r或者run ——运行程序

start ——开始逐步调试

n或者next ——下一步，遇到函数直接得到函数运行结果，并不跟踪进入函数

s或者step ——下一步，遇到函数跟踪进入函数

p 变量名 ——查看变量数据

l或者list 行号 ——查看指定行(默认当前调试行)附近代码(上下5行)

until 行号 ——直接运行到指定行

b(break) 行号 ——向当前调试文件的指定行添加断点

b 函数名 ——向指定函数开头添加断点

b 文件名:行号 ——向指定文件指定行添加断点

watch 变量名 ——变量监控，给变量添加断点，当变量内容发生改变，停止

i(info) b(breakpoints) ——查看所有断点信息

d(delete) 序列号 ——删除指定断点，默认删除所有

c(continue) ——继续运行

bt(breaktrace) ——查看函数调用栈信息

q(quit) ——退出gdb

make/Makefile：项目自动化构建工

Makefile：定义编译规则的普通文件

make：是一个解释程序，解释执行Makefile中定义的编译规则

执行make命令，这时候make程序会在当亲目录下寻找Makefile文件，解释其中的编译规则

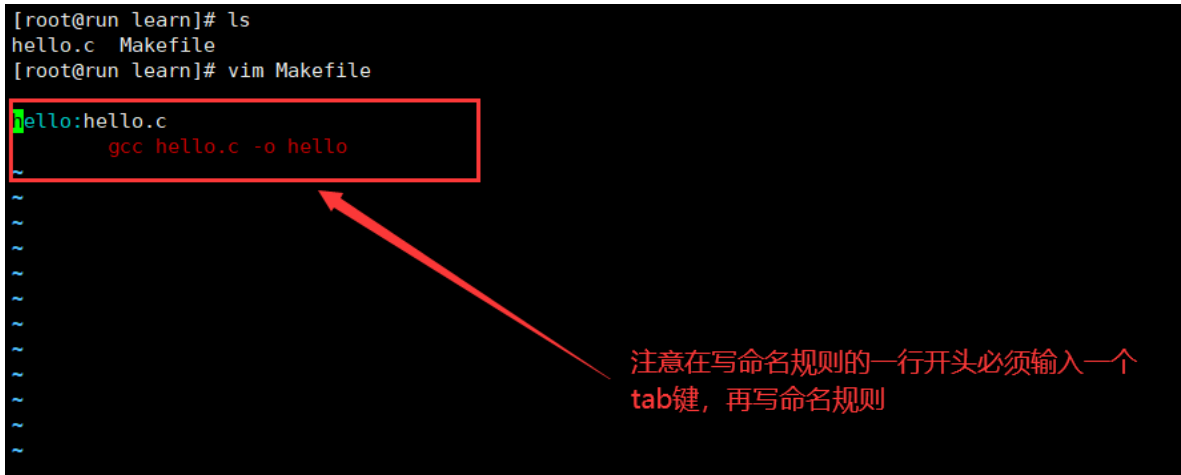
```
[root@run learn]# ls
hello.c  Makefile
[root@run learn]# vim Makefile
[root@run learn]# make
gcc hello.c -o hello
[root@run learn]# ls
hello hello.c Makefile
[root@run learn]#
```

make命令找到Makefile文件，并执行生成可执行文件

编写规则

- 1 目标对象:依赖对象
- 2
- 3 规则命令(前面有一个tab键)

```
[root@run learn]# ls
hello.c  Makefile
[root@run learn]# vim Makefile
```



hello:hello.c
gcc hello.c -o hello

注意在写命名规则的一行开头必须输入一个tab键，再写命名规则

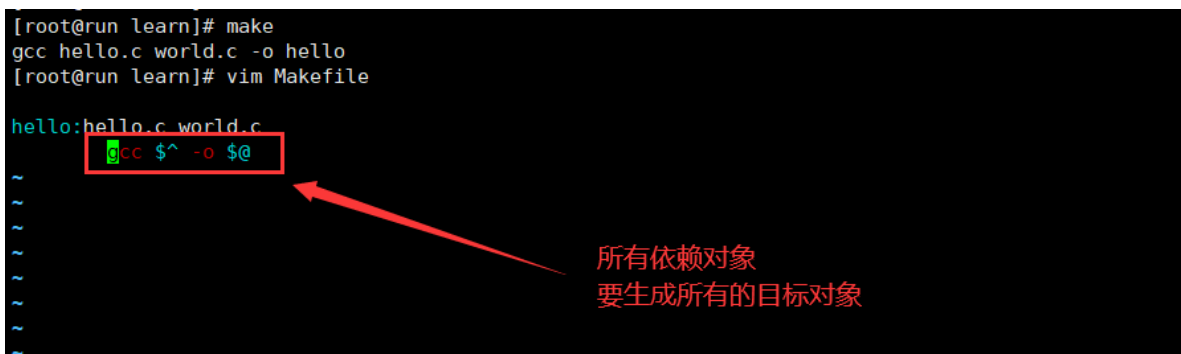
执行规则

- 通过目标对象与依赖对象最后一次修改时间，判断目标对象是否需要重新生成
- make在Makefile中只找第一个目标对象，为了生成这个目标对象，而执行命令，完成之后直接退出(后边的对象都不再生成)
- make在Makefile中找到第一个对象，这时候如果这个对象的依赖不存在，则在后续编译规则中，寻找是否可以生成这个依赖对象，当所有的依赖对象生成完毕后，最终生成目标对象

预定义变量

- \$@ ——要生成所有的目标对象
- \$^ ——所有的依赖对象
- \$< ——依赖对象中的第一个

```
[root@run learn]# make
gcc hello.c world.c -o hello
[root@run learn]# vim Makefile
```



hello:hello.c world.c
gcc \$^ -o \$@

所有依赖对象
要生成所有的目标对象