概述

什么是makefile？或许很多Windows的程序员都不知道这个东西，因为那些Windows的集成开发环境（integrated development environment，IDE）都为你做了这个工作，但我觉得要作一个好的和专业的程序员，makefile还是要懂。这就好像现在有这么多的HTML编辑器，但如果你想成为一个专业人士，

你还是要了解HTML的标签的含义。特别在Unix下的软件编译，你就不能不自己写makefile了，会不会写makefile，从一个侧面说明了一个人是否具备完成大型工程的能力。因为，makefile关系到了整个工程的编译规则。一个工程中的源文件不计其数，并且按类型、功能、模块分别放在若干个目录中，makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为makefile就像一个Shell脚本一样，其中也可以执行操作系统的命令。

makefile带来的好处就是——“自动化编译”，一旦写好，只需要一个make命令，整个工程完全自动编译，极大的提高了软件开发的效率。 make是一个命令工具，是一个解释makefile中指令的命令工具，一般来说，大多数的IDE都有这个命令，比如：Delphi的make，Visual C++的nmake，Linux下GNU的make。可见，makefile都成为了一种在工程方面的编译方法。

现在讲述如何写makefile的文章比较少，这是我想写这篇文章的原因。当然，不同产商的make各不相同，也有不同的语法，但其本质都是在 “文件依赖性”上做文章，这里，我仅对GNU的make进行讲述，我的环境是RedHat Linux 8.0，make的版本是3.80。毕竟，这个make是应用最为广泛的，也是用得最多的。而且其还是最遵循于IEEE 1003.2-1992标准的（POSIX.2）。

在这篇文档中，将以C/C++的源码作为基础，所以必然涉及一些关于C/C++的编译的知识。关于这方面的内容，还请各位查看相关的编译器的文档。这里所默认的编译器是UNIX下的GCC和CC。

关于程序的编译和链接

在此，我想多说关于程序编译的一些规范和方法。一般来说，无论是C还是C++，首先要把源文件编译成中间代码文件，在Windows下也就是“.obj” 文件，UNIX下是 “.o” 文件，即Object File，这个动作叫做编译（compile）。然后再把大量的Object File合成可执行文件，这个动作叫作链接（link）。

编译时，编译器需要的是语法的正确，函数与变量的声明的正确。对于后者，通常是你需要告诉编译器头文件的所在位置（头文件中应该只是声明，而定义应该放在C/C++文件中），只要所有的语法正确，编译器就可以编译出中间目标文件。一般来说，每个源文件都应该对应于一个中间目标文件（.o 文件或

.obj 文件）。

链接时，主要是链接函数和全局变量。所以，我们可以使用这些中间目标文件（.o 文件或.obj 文件）来链接我们的应用程序。链接器并不管函数所在的源文件，只管函数的中间目标文件（Object File），在大多数时候，由于源文件太多，编译生成的中间目标文件太多，而在链接时需要明显地指出中间目标文件名，这对于编译很不方便。所以，我们要给中间目标文件打个包，Windows下这种包叫“库文件”（Library File），也就是.lib 文件，在UNIX下，是Archive File，也就是.a文件。

总结一下：

1. 源文件首先会生成中间目标文件。源文件编译生成机器语言Object File（Linux为.o文件、Windows为.obj文件）在编译时，编译器只检测程序语法和函数、变量是否被声明。如果函数未被声明，编译器会给出一个警告，但可以生成Object File。
2. 再由中间目标文件链接生成执行文件。而在链接程序时，链接器会在所有的Object File中找寻函数的实现，如果找不到，那到就会报链接错误码

（Linker Error），在VC下，这种错误一般是： Link 2001错误，意思说是说，链接器未能找到函数的实现。你需要指定函数的Object File。

Make file简介

这里要说明一点的是， clean 不是一个文件，它只不过是一个动作名字，有点像c语言中的label一样，其冒号后什么也没有，那么，make就不会自动去找它的依赖性，也就不会自动执行其后所定义的命令。要执行其后的命令，就要在make命令后明显得指出这个label的名字。这样的方法非常有用，我们可以在一个makefile中定义不用的编译或是和编译无关的命令，比如程序的打包，程序的备份，等等。

Makefile中的换行符

\

Makefile中的变量

“变量名” = “需要指代的变量”

使用： $（变量名）

Makefile中的自动推导

GNU的make很强大，它可以自动推导文件以及文件依赖关系后面的命令，于是我们就没必要去在每一个 .o 文件后都写上类似的命令，因为，我们的make会自动识别，并自己推导命令。

只要make看到一个 .o 文件，它就会自动的把 .c 文件加在依赖关系中，并且 g++ -c whatever.c 也会被推导出来，于是，我们的makefile再也不用写得这么复杂。我们的新makefile又出炉了。

例：

t1.o: t1.cpp

g++ -c t1.cpp

可改写为：

t1.o:

makefile清空规则

前面说过， .PHONY 表示 clean 是一个“伪目标”。而在 rm 命令前面加了一个小减号的意思就是，也许某些文件出现问题，但不要管，继续做后面的事。当然， clean 的规则不要放在文件的开头，不然，这就会变成make的默认目标，相信谁也不愿意这样。不成文的规矩是——“clean从来都是放在文件的最后”。

.PHONY: clean

clean:

-rm \*.o

Makefile注释

#

转义\#

Makefile的文件名

一般用“Makefile”。“makefile”

你可以使用别的文件名来书写Makefile，比如：“Make.Linux”，“Make.Solaris”，“Make.AIX”等，如果要指定特定的Makefile，你可以使用make的 -f 和 --file 参数，如： make -f Make.Linux 或 make --file Make.AIX 。