GNU make文章阅读笔记：

一.概述：

在Makefile文件中描述了整个工程所有文件的编译顺序、编译规则

通常，一个高级语言的源文件都可对应一个目标文件。目标文件在Linux中默认后缀为“.o”

二.GNU make介绍：

make在执行时，需要一个命名为Makefile的文件。这个文件告诉make以何种方式编译源代码和链接程序

make会自动根据修改情况完成源文件的对应.o文件的更新、库文件的更新、最终的可执行程序的更新

规则:

一个简单的Makefile描述规则组成：

***TARGET... : PREREQUISITES...***

***COMMAND***

***...***

***...***

target：规则的目标。通常是最后需要生成的文件名或者为了实现这个目的而必需的中间过程文件名。可以是.o文件、也可以是最后的可执行程序的文件名等。另外，目标也可以是一个make执行的动作的名称，如目标“clean”，我们称这样的目标是“伪目标”。

prerequisites：规则的依赖。生成规则目标所需要的文件名列表。通常一个目标依赖于一个或者多个文件。

command：规则的命令行。是规则所要执行的动作（任意的shell命令或者是可在shell下执行的程序）。它限定了make执行这条规则时所需要的动作。**[Tab]**字符告诉**make**此行是一个命令行

Makefile中“规则”就是描述在什么情况下、如何重建规则的目标文件，通常规则中包括了目标的依赖关系（目标的依赖文件）和重建目标的命令。

书写时’\’,续行符,后不能有空格

当规则的目标是一个文件，在它的任何一个依赖文件被修改以后，在执行“make”时这个目标文件将会被重新编译或者重新连接

make如何工作:

默认的情况下，make执行的是Makefile中的第一个规则，此规则的第一个目标称之为“最终目的”或者“终极目标”

Makefile的执行过程:

对于一个Makefile文件，“make”首先解析终极目标所在的规则,根据其依赖文件依次寻找创建这些依赖文件的规则。首先为第一个依赖文件寻找创建规则，如果第一个依赖文件依赖于其它文件，则同样为这个依赖文件寻找创建规则……，直到为所有的依赖文件找到合适的创建规则。之后make从最后一个规则（上例目标为main.o的规则）回退开始执行，最终完成终极目标的第一个依赖文件的创建和更新。之后对第二个、第三个、第四个……终极目标的依赖文件执行同样的过程.

变量:

在makefile中可以使用变量,变量定义的时候如一般的定义模式

当我们引用变量的时候方式为: $(变量名)

书写规则:单目标,多依赖

伪目标:

***.PHONY : clean***

***clean :***

***-rm edit $(objects)***

1. 通过“.PHONY”特殊目标将“clean”目标声明为伪目标。避免当磁盘上存在一个名为“clean”文件时，目标“clean”所在规则的命令无法执行

2. 在命令行之前使用“-”，意思是忽略命令“rm”的执行错误

三.makefile总述

组成:

一个完整的Makefile中，包含了5个东西：显式规则、隐含规则、变量定义、指示符和注释

􀂗 显式规则：它描述了在何种情况下如何更新一个或者多个被称为目标的文件（Makefile的目标文件）。书写Makefile时需要明确地给出目标文件、目标的依赖文件列表以及更新目标文件所需要的命令（有些规则没有命令，这样的规则只是纯粹的描述了文件之间的依赖关系）。

􀂗 隐含规则：它是make根据一类目标文件（典型的是根据文件名的后缀）而自动推导出来的规则。make根据目标文件的名，自动产生目标的依赖文件并使用默认的命令来对目标进行更新（建立一个规则）

􀂗 变量定义：使用一个字符或字符串代表一段文本串，当定义了一个变量以后，Makefile后续在需要使用此文本串的地方，通过引用这个变量来实现对文本串的使用。

􀂗 Makefile指示符：指示符指明在make程序读取makefile文件过程中所要执行的一个动作。其中包括：

􀂗 读取一个文件，读取给定文件名的文件，将其内容作为makefile文件的一部分。

􀂗 决定（通常是根据一个变量的得值）处理或者忽略Makefile中的某一特定部分。

􀂗 定义一个多行变量。

􀂗 注释：Makefile中“#”字符后的内容被作为是注释内容（和shell脚本一样）处理。如果此行的第一个非空字符为“#”，那么此行为注释行。注释行的结尾如果存在反斜线（\），那么下一行也被作为注释行。一般在书写Makefile时推荐将注释作为一个独立的行，而不要和Makefile的有效行放在一行中书写。当在Makefile中需要使用字符“#”时，可以使用反斜线加“#”（\#）来实现（对特殊字符“#”的转义），其表示将“#”作为一字符而不是注释的开始标志。

make会在工作目录（执行make的目录）下按照文件名顺序寻找makefile文件读取并执行，查找的文件名顺序为：“GNUmakefile”、“makefile”、“Makefile”。如果make程序在工作目录下无法找到以上三个文件中的任何一个，它将不读取任何其他文件作为解析对象

通过make的“-f”或者“--file”选项来指定make读取的makefile文件

Include:

“include”指示符告诉make暂停读取当前的Makefile，而转去读取“include”指定的一个或者多个文件，完成以后再继续当前Makefile的读取

***include FILENAMES...***

GUN make的执行过程:

第一阶段：读取所有的makefile文件（包括“MAKIFILES”变量指定的、指示符“include”指定的、以及命令行选项“-f(--file)”指定的makefile文件），内建所有的变量、明确规则和隐含规则，并建立所有目标和依赖之间的依赖关系结构链表。

在第二阶段：根据第一阶段已经建立的依赖关系结构链表决定哪些目标需要更新，并使用对应的规则来重建这些目标。

make的执行过程如下：

1. 依次读取变量“MAKEFILES”定义的makefile文件列表

2. 读取工作目录下的makefile文件（根据命名的查找顺序“GNUmakefile”，“makefile”，“Makefile”，首先找到那个就读取那个）

3. 依次读取工作目录makefile文件中使用指示符“include”包含的文件

4. 查找重建所有已读取的makefile文件的规则（如果存在一个目标是当前读取的某一个makefile文件，则执行此规则重建此makefile文件，完成以后从第一步开始重新执行）

5. 初始化变量值并展开那些需要立即展开的变量和函数并根据预设条件确定执行分支

6. 根据“终极目标”以及其他目标的依赖关系建立依赖关系链表

7. 执行除“终极目标”以外的所有的目标的规则（规则中如果依赖文件中任一个文件的时间戳比目标文件新，则使用规则所定义的命令重建目标文件）

8. 执行“终极目标”所在的规则

四.Makefile的规则

Wildcard:

在变量的定义和函数引用时，通配符将失效。这种情况下如果需要通配符有效，就需要使用函数“wildcard”，它的用法是：$(wildcard PATTERN...) 。在Makefile中，它被展开为已经存在的、使用空格分开的、匹配此模式的所有文件列表。如果不存在任何符合此模式的文件，函数会忽略模式字符并返回空。

目录搜索:

VPATH:

变量“VPATH”可以指定依赖文件的搜索路径，当规则的依赖文件在当前目录不存在时，make会在此变量所指定的目录下去寻找这些依赖文件,为Makefile中所有文件的搜索路径，包括了规则的依赖文件和目标文件.

***VPATH = src:../headers***

vpath:

1、***vpath PATTERN DIRECTORIES***

为所有符合模式“PATTERN”的文件指定搜索目录“DIRECTORIES”。多个目录使用空格或者冒号（：）分开。类似上一小节的“VPATH”变量。

2、***vpath PATTERN***

清除之前为符合模式“PATTERN”的文件设置的搜索路径。

3、***vpath***

清除所有已被设置的文件搜索路径。

make在解析Makefile文件执行规则时对文件路径保存或废弃所依据的算法如下：

1. 首先，如果规则的目标文件在Makefile文件所在的目录（工作目录）下不存在，那么就执行目录搜寻。

2. 如果目录搜寻成功，在指定的目录下存在此规则的目标。那么搜索到的完整的路径名就被作为临时的目标文件被保存。

3. 对于规则中的所有依赖文件使用相同的方法处理。

4. 完成第三步的依赖处理后，make程序就可以决定规则的目标是否需要重建，两种情况时后续处理如下：

a) 规则的目标不需要重建：那么通过目录搜索得到的所有完整的依赖文件路径名有效，同样，规则的目标文件的完整的路径名同样有效。就是说，当规则的目标不需要被重建时，规则中的所有的文件完整的路径名有效。已经存在的目标文件所在的目录不会被改变。

b) 规则的目标需要重建：那么通过目录搜索所得到的目标文件的完整的路径名无效，规则中的目标文件将会被在工作目录下重建。就是说，当规则的目标需要重建时，规则的目标文件会在工作目录下被重建，而不是在目录搜寻时所得到的目录。这里，必须明确：此种情况只有目标文件的完整路径名失效，依赖文件的完整路径名是不会失效的。否则将无法重建目标。

Makefile伪目标:

使用伪目标有两点原因：1. 避免在我们的Makefile中定义的只执行命令的目标（此目标的目的为了执行执行一些列命令，而不需要创建这个目标）和工作目录下的实际文件出现名字冲突。2. 提高执行make时的效率

一个伪目标不作为另外一个目标的依赖

强制目标:(FORCE)

***clean: FORCE***

***rm $(objects)***

***FORCE:***

在执行此规则时，目标总会被认为是最新的.这样的目标在作为一个规则的依赖时，因为依赖总被认为被更新过，因此作为依赖所在的规则中定义的命令总会被执行

空目标文件只是用来记录上一次执行此规则命令的时间

Makefile的特殊目标 (个人粘过来作为查询参考)

**.PHONY**：

目标“**.PHONY**”的所有的依赖被作为伪目标。伪目标时这样一个目标：当使用make命令行指定此目标时，这个目标所在规则定义的命令、无论目标文件是否存在都会被无条件执行。

**.SUFFIXES:**

特殊目标“SUFFIXES”的所有依赖指出了一系列在后缀规则中需要检查的后缀名（就是当前make需要处理的后缀）。

**.DEFAULT**

Makefile中，目标“.DEFAULT”所在规则定义的命令，被用在重建那些没有具体规则的目标（明确规则和隐含规则）。就是说一个文件作为某个规则的依赖，但却不是另外一个规则的目标时。Make程序无法找到重建此文件的规则，此种情况时就执行“.DEFAULT”所指定的命令。

**.PRECIOUS**

目标“.PRECIOUS”的所有依赖文件在make过程中会被特殊处理：当命令在执行过程中被中断时，make不会删除它们。而且如果目标的依赖文件是中间过程文件，同样这些文件不会被删除。这一点目标“.PRECIOUS”和目标“.SECONDAY”实现的功能相同。

另外，目标“.PRECIOUS”的依赖文件也可以是一个模式，例如“%.o”。这样可以保留有规则创建的中间过程文件。

**.INTERMEDIATE**

目标“.INTERMEDIATE”的依赖文件在make时被作为中间过程文件对待。没有任何依赖文件的目标“.INTERMEDIATE”没有意义。

**.SECONDARY**

目标“.SECONDARY”的依赖文件被作为中间过程文件对待。但这些文件不会被自动删除

没有任何依赖文件的目标“.SECONDARY”的含义是：将所有的文件作为中间过程文件（不会自动删除任何文件）。

**.DELETE\_ON\_ERROR**

如果在Makefile中存在特殊目标“.DELETE\_ON\_ERROR”，make在执行过程中，如果规则的命令执行错误，将删除已经被修改的目标文件。

**.IGNORE**

如果给目标“.IGNORE”指定依赖文件，则忽略创建这个文件所执行命令的错误。给此目标指定命令是没有意义的。当此目标没有依赖文件时，将忽略所有命令执行的错误。

**.LOW\_RESOLUTION\_TIME**

目标“.LOW\_RESOLUTION\_TIME”的依赖文件被make认为是低分辨率时间戳文件。给目标“.LOW\_RESOLUTION\_TIME”指定命令是没有意义的。

通常文件的时间辍都是高分辨率的，make在处理依赖关系时、对规则目标-依赖文件的高分辨率的时间戳进行比较，判断目标是否过期。但是在系统中并没有提供一个修改文件高分辨率时间辍的机制（方式），因此类似“cp -p”这样的命令在根据源文件创建目的文件时，所产生的目的文件的高分辨率时间辍的细粒度部分被丢弃（来源于源文件）。这样可能会造成目的文件的时间戳和源文件的相等甚至不及源文件新。处理此类命令创建的文件时，需要将命令创建的文件作为目标“.LOW\_RESOLUTION\_TIME”的依赖，声明这个文件是一个低分辨率时间辍的文件。例如：

***.LOW\_RESOLUTION\_TIME: dst***

***dst: src***

***cp -p src dst***

首先规则的命令“cp –p src dst”，所创建的文件“dst”在时间戳上稍稍比“src”晚（因为命令不能更新文件“dst”的细粒度时间）。因此make在判断文件依赖关系时会出现误判，将文件作为目标“.LOW\_RESOLUTION\_TIME”的依赖后，只要规则中目标和依赖文件的时间戳中的初始时间相等，就认为目标已经过期。这个特殊的目标主要作用是，弥补系统在没有提供修改文件高分辨率时间戳机制的情况下，某些命令在make中的一些缺陷。

**.SILENT**

出现在目标“.SILENT”的依赖列表中的文件，make在创建这些文件时，不打印出重建此文件所执行的命令。同样，给目标“.SILENT”指定命令行是没有意义的。

没有任何依赖文件的目标“.SILENT”告诉make在执行过程中不打印任何执行的命令。现行版本make支持目标“.SILENT”的这种功能和用法是为了和旧版本的兼容。在当前版本中如果需要禁命令执行过程的打印，可以使用make的命令行参数“-s”或者“--silent”。参考9.7 make的命令行选项一节

**.EXPORT\_ALL\_VARIABLES**

此目标应该作为一个简单的没有依赖的目标，它的功能含义是将之后所有的变量传递给子make进程。参考5.6 make的递归执行一节

**.NOTPARALLEL**

Makefile中，如果出现目标“.NOPARALLEL”，则所有命令按照串行方式执行，即使存在make的命令行参数“-j”。但在递归调用的字make进程中，命令可以并行执行。此目标不应该有依赖文件，所有出现的依赖文件将被忽略。

五.规则的命令

如果没有明确指定，那么对所有规则中命令行的解析使用“/bin/sh”来完成

命令回显:

make在执行命令行之前会把要执行的命令行输出到标准输出设备

如果规则的命令行以字符“@”开始，则make在执行这个命令时就不会回显这个将要被执行的命令.

另外，如果使用make的命令行参数“-n”或“--just-print”，那么make执行时只显示所要执行的命令，但不会真正的去执行这些命令

make参数“-s”或“--slient”则是禁止所有执行命令的显示

命令的执行:

***foo : bar/lose***

***cd bar; gobble lose > ../foo***

在一个规则的命令中，命令行“**cd**”改变目录不会对其后的命令的执行产生影响。就是说其后的命令执行的工作目录不会是之前使用“**cd**”进入的那个目录。如果要实现这个目的，就不能把“**cd**”和其后的命令放在两行来书写。而应该把这两条命令写在一行上，用分号分隔

‘-’用来告诉make忽略命令的执行失败.

中断make执行:

make在执行命令时如果收到一个致命信号（终止make），那么make将会删除此过程中已经重建的那些规则的目标文件。其依据是此目标文件的当前时间戳和make开始执行时此文件的时间戳是否相同。

删除这个目标文件的目的是为了确保下一次make时目标文件能够被正确重建

make的递归过程指的是：在Makefile中使用“make”作为一个命令来执行本身或者其它makefile文件的过程

九.执行make

所有通过“-f”参数指定的文件都被作为make解析执行的makefile文件

例如：“make –f altmake”

Makefile书写时需要保证：第一个目标的编译规则就描述了整个工程或者程序的编译过程和规则。如果Makefile的第一个规则有多个目标，那么默认的终极目标是多个目标中的第一个。我们在Makefile所在的目录下执行“make”时，将完成对默认终极目标的重建。

make 所支持的命令行参数:手册 P138

自动化变量:

**$%**

当规则的目标文件是一个静态库文件时，代表静态库的一个成员名。例如，规则的目标是“foo.a(bar.o)”，那么，“$%”的值就为“bar.o”，“$@”的值为“foo.a”。如果目标不是静态库文件，其值为空。

**$<**

规则的第一个依赖文件名。如果是一个目标文件使用隐含规则来重建，则它代表由隐含规则加入的第一个依赖文件。

**$?**

所有比目标文件更新的依赖文件列表，空格分割。如果目标是静态库文件名，代表的是库成员（.o文件）。

**$^**

规则的所有依赖文件列表，使用空格分隔。如果目标是静态库文件，它所代表的只能是所有库成员（.o文件）名。一个文件可重复的出现在目标的依赖中，变量“$^”只记录它的一次引用情况。就是说变量“$^”会去掉重复的依赖文件。

**$+**

类似“$^”，但是它保留了依赖文件中重复出现的文件。主要用在程序链接时库的交叉引用场合。

**$\***

在模式规则和静态模式规则中，代表“茎”。“茎”是目标模式中“%”所代表的部分（当文件名中存在目录时，“茎”也包含目录（斜杠之前）部分。例如：文件“dir/a.foo.b”，当目标的模式为“a.%.b”时，“$\*”的值为“dir/a.foo”。“茎”对于构造相关文件名非常有用。

自动化变量“$\*”需要两点说明：

􀂾对于一个明确指定的规则来说不存在茎，这种情况下$\*”的含义发生改变。此时，如果目标文件名带有一个可识别的后缀（参考10.7 后缀规则一节），那么“$\*”表示文件中除后缀以外的部分。例如：“foo.c”则“$\*”的值为：“foo”，因为.c是一个可识别的文件后缀名。GUN make对明确规则的这种奇怪的处理行为是为了和其它版本的make兼容。通常，在除静态规则和模式规则以外，明确指定目标文件的规则中应该避免使用这个变量。

􀂾当明确指定文件名的规则中目标文件名包含不可识别的后缀时，此变量为空。