Makefile 教程

- 感谢徐海兵翻译
- 主讲人：秦风岭

Makefile 的内容

- 在一个完整的 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。
- 举例：

  default_ovtc.mak,makefile

- 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。
- 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。
- 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 :

  1. 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。
  2. 决定处理或者忽略 Makefile 中的某一特定部分。
  3. 定义一个多行变量。
- 注 :Makefile 中“ #” 字符后的内容被作为注释内容处理。

makefile 文件的命名

- 默认的情况下 ,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” 。

  推荐使用“ Makefile”, 首字母大写而比较显著，一般和当前目录的一些重要文件 (README,Chagelist 等) 靠近 , 在寻找时会比较容易的发现它。而

  “GNUmakefile” 是我们不推荐使用的文件名 , 因为以此命名的文件只有“ GNU make” 才可以识别 , 而其他版本的 make 程序只会在工作目录下查找“ makefile” 和“ Makefile” 这两个文件。当 makefile 文件的命名不是这三个任何一个时 , 需要通过 make 的“ -f” 或者“ --file” 选项来指定 make 读取的 makefile 文件。给 make 指定 makefile 文件的格式为 :

  “-f NAME” 或者 “ --file=NAME” 。

包含其它 makefile 文件

- “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。 Makefile 中指示符“ include” 书写在独立的一行 , 其形式如下 :

  include FILENAMES...

  FILENAMES 是 shell 所支持的文件名 (可以使用通配符)。

  指示符“ include” 所在的行可以一个或者多个空格

  些空格) 开始 , 切忌不能以 [Tab] 字符开始 (如果一行以 [Tab] 字符开始 make 程序将此行作为一个命令行来处理)。指示符“ include” 和文件名之间使用空格或者 [Tab] 键隔开。行尾的空白字符在处理时被忽略。使用指示符包含进来的 Makefile 中 , 如果存在变量或者函数的引用。它们将会在包含它们的 Makefile 中被展开。

- 通常指示符“ include” 用在以下场合 :

  1. 有多个不同的程序 , 由不同目录下的几个独立的 Makefile 来描述其重建规则。它们需要使用一组通用的变量定义或者模式规则。通用的做法是将这些共同使用的变量或者模式规则定义在一个文件中 (没有具体的文件命名限制), 在需要使用的 Makefile 中使用指示符“ include” 来包含此文件。
  2. 当根据源文件自动产生依赖文件时 ; 我们可以将自动产生的依赖关系保存在另外一个文件中 , 主 Makefile 使用指示符“ include” 包含这些文件。

- 如果指示符“ include ”指定的文件不是以斜线开始（绝对路径，如 /usr/src/Makefile...）而且当前目录下也不存在此文件 ;make 将根据文件名试图在以下几个目录下查找 : 首先 , 查找使用命令行选项“ -I” 或者“ --include-dir” 指定的目录【注：和 gcc 的选项要区分】, 如果找到指定的文件 , 则使用这个文件 ; 否则继续依此搜索以下几个目录 (如果其存在)
“/usr/gnu/include” 、“ /usr/local/include” 和、“ /usr/include” 。当在这些目录下都没有到“ include” 指定的文件时 ,make 将会提示一个包含文件未找到的告警提示 , 但是不会立刻退出。而是继续处理 Makefile 的后续内容。当完成读取整个 Makefile 后 ,make 将试图使用规则来创建通过指示符“ include” 指定的但未找到的文件 , 当不能创建它时 ,make 将提示致命错误并退出。会输出类似如下错误提示 :

  Makefile: 错误的行数 : 未找到文件名 :

  提示信息 (No such file or directory)

  Make: *** No rule to make target ‘<filename>’. Stop

- 通常我们在 Makefile 中可使用“ -include” 来代替“ include”, 来忽略由于包含文件不存在或者无法创建时的错误提示 (“-” 的意思是告诉 make, 忽略此操作的错误。 make 继续执行)。像下边那样 :

  -include FILENAMES...

  使用这种方式时 , 当所要包含的文件不存在时不会有错误提示、 make 也不会退出。
- 为了和其它的 make 程序进行兼容。也可以使用“ sinclude” 来代替“ -include”(GNU 所支持的方式)。

变量 MAKEFILES

- 如果在当前环境定义了一个“ MAKEFILES” 环境变量 ,make 执行时首先将此变量的值作为需要读入的 Makefile 文件 , 多个文件之间使用空格分开。 通过这种方式设置的“隐含规则”和定义的变量可以被任何 make 进程使用。 (有点象 C 语言中的全局变量) 。

  推荐的做法 : 在需要包含其它 makefile 文件时使用指示符“ include” 来实现。
- 示例：

  文件： Makefile

变量 MAKEFILE_LIST

- make 程序在读取多个 makefile 文件时，包括由环境变量“ MAKEFILES”指定、命令行指定、当前工作目录下的默认的以及使用指示符“ include”指定包含的，在对这些文件进行解析执行之前 make 读取的文件名将会被自动依次追加到变量“ MAKEFILE_LIST”的定义域中。这样我们就可以通过测试此变量的最后一个字来获取当前 make 程序正在处理的 makefile 文件名。
- 示例：

  文件： Makefile

makefile 文件的重建

- Makefile 可由其它文件生成。如果 Makefile 由其它文件重建，那么在 make 在开始解析这个 Makefile 时需要重新读取更新后的 Makefile、而不是之前的 Makefile。

其他特殊变量

- GNU make 支持一个特殊的变量，此变量不能通过任何途经给它赋值。它被展开为一个特定的值。一个重要的特殊的变量是“ .VARIABLES”。它被展开以后是此引用点之前、 makefile 文件中所定义的所有全局变量列表。包括 : 空变量 (未赋值的变量) 和 make 的内嵌变量，但不包含目标指定的变量，目标指定变量值在特定目标的上下文有效。
- 示例：

  文件： Makefile

  命令： make testvariables

  一般人绝对看不懂！

make 如何解析 makefile 文件

- GUN make 的执行过程分为两个阶段。

  第一阶段 : 读取所有的 makefile 文件 (包括“ MAKEFILES” 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。

Makefile 的执行过程总结

- 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

make 的内嵌函数

- make 的函数

  GNU make 的函数提供了处理文件名、变量、文本和命令的方法。使用函数我们的 Makefile 可以书写的更加灵活和健壮。可以在需要的地方地调用函数来处理指定的文本 (需要处理的文本作为函数的参数), 函数的在调用它的地方被替换为它的处理结果。函数调用 (引用) 的展开和变量引用的展开方式相同。

函数的调用语法

- GNU make 函数的调用格式类似于变量的引用，以“ $” 开始表示一个引用。语法格式如下：

  $(FUNCTION ARGUMENTS)

  或者：

  ${FUNCTION ARGUMENTS}

文本处理函数 (部分介绍)

- $(subst FROM,TO,TEXT)

  函数名称 : 字符串替换函数— subst 。

  函数功能 : 把字串“ TEXT” 中的“ FROM” 字符替换为“ TO” 。

  返回值 : 替换后的新字符串。

  示例 :

  $(subst code,/home/user/project/code,code/program.c)

  替换“ code/program.c” 中的“ code” 为“ /home/user/project/code” 结果得到字符串中的为“ /home/user/project/code/program.c” 。

- 示例:

  文件: Makefile

  命令: make substfun

- $(patsubst PATTERN,REPLACEMENT,TEXT)

  函数名称 : 模式替换函数— patsubst 。

  函数功能 : 搜索“ TEXT” 中以空格分开的单词 , 将符合模式“ PATTERN” 替换为“ REPLACEMENT” 。参数“ PATTERN” 中可以使用模式通配符“ %” 来代表一个单词中的若干字符。如果参数“ REPLACEMENT” 中也包含一个“ %”, 那么“ REPLACEMENT” 中的“ %” 将是“ PATTERN” 中的那个“ %” 所代表的字符串。

  返回值 : 替换后的新字符串。

  函数说明 : 参数“ TEXT” 单词之间的多个空格在处理时被合并为一个空格 , 并忽略前导和结尾空格。

- 简化版 $(VAR:PATTERN=REPLACEMENT)

  示例：

  文件： Makefile

  命令： make patsubstfun

- $(strip STRINT)

  函数名称 : 去空格函数— strip 。

  函数功能 : 去掉字串 (若干单词 , 使用若干空字符分割)“STRINT” 开头和结尾的空字符 , 并将其中多个连续空字符合并为一个空字符。

  返回值 : 无前导和结尾空字符、使用单一空格分割的多单词字符串。

  函数说明 : 空字符包括空格、 [Tab] 等不可显示字符。

  示例 :

  文件： Makefile

  命令： make

- $(sort LIST)

  函数名称 : 排序函数— sort 。

  函数功能 : 给字串“ LIST” 中的单词以首字母为准进行排序 (升序), 并取掉重复的单词。

  返回值 : 空格分割的没有重复单词的字串。

  函数说明 : 两个功能 , 排序和去字串中的重复单词。可以单独使用其中一个功能。

  示例 :

  文件： Makefile

  命令： make sortfun

文件名处理函数（部分介绍）

- 主要用来对一系列空格分割的文件名进行转换，这些函数的参数被作为若干个文件名来对待。函数对作为参数的一组文件名按照一定方式进行处理并返回空格分割的多个文件名序列。

- $(dir NAMES...)

  函数名称 : 取目录函数— dir 。

  函数功能 : 从文件名序列“ NAMES...” 中取出各个文件名的目录部分。文件名的目录部分就是包含在文件名中的最后一个斜线 (“/”)(包括斜线) 之前的部分。

  返回值 : 空格分割的文件名序列“ NAMES...” 中每一个文件的目录部分。函数说明 : 如果文件名中没有斜线 , 认为此文件为当前目录 (“./”) 下的文件。

  示例 :

  文件： Makefile

  命令： make dirfun

- $(notdir NAMES...)

  函数名称 : 取文件名函数—— notdir 。

  函数功能 : 从文件名序列“ NAMES...” 中取出非目录部分。目录部分是指最后一个斜线 (“/”)(包括斜线) 之前的部分。删除所有文件名中的目录部分 , 只保留非目录部分。

  返回值 : 文件名序列“ NAMES...” 中每一个文件的非目录部分。

  函数说明 : 如果“ NAMES...” 中存在不包含斜线的文件名 , 则不改变这个文件名。以反斜线结尾的文件名 , 是用空串代替 , 因此当“ NAMES...” 中存在多个这样的文件名时 , 返回结果中分割各个文件名的空格数目将不确定 ! 这是此函数的一个缺陷。

  示例 :

  文件： Makefile

  命令： make notdirfun

- $(addsuffix SUFFIX,NAMES...)

  函数名称 : 加后缀函数— addsuffix 。

  函数功能 : “NAMES...” 为中的每一个文件名添加后缀“ SUFFIX” 参数。 “ NAMES...” 为空格分割的文件名序列 , 将“ SUFFIX” 追加到此序列的每一个文件名的末尾。

  返回值 : 以单空格分割的添加了后缀“ SUFFIX” 的文件名序列。

  函数说明 :

  示例 :

  文件： Makefile

  命令： make addsuffixfun

- $(addprefix PREFIX,NAMES...)

  函数名称 : 加前缀函数— addprefix 。 函数功能 : “NAMES...” 为中的每一个文件名添加前缀“ PREFIX” 参数。 “ NAMES...” 是空格分割的文件名序列 , 将“ SUFFIX” 添加到此序列的每一个文件名之前。

  返回值 : 以单空格分割的添加了前缀“ PREFIX” 的文件名序列。

  函数说明 :

  示例 :

  文件： Makefile

  命令： make addprefixfun

- $(wildcard PATTERN)

  函数名称 : 获取匹配模式文件名函数— wildcard

  函数功能 : 列出当前目录下所有符合模式“ PATTERN” 格式的文件名。

  返回值 : 空格分割的、存在当前目录下的所有符合模式“ PATTERN” 的文件名。

  函数说明 :

  “PATTERN” 使用 shell 可识别的通配符 , 包括“ ?” (单字符)、“ *” (多字符) 等。

  示例 :

  文件 :Makefile

  命令 :make wildcardfun

shell 函数

- 函数功能 : 函数“ shell” 所实现的功能和 shell 中的引用 (``) 相同。实现对命令的扩展。

  返回值 : 函数“ shell” 的参数 (一个 shell 命令) 在 shell 环境中的执行结果。

  函数说明 : 函数本身的返回值是其参数的执行结果 , 没有进行任何处理。对结果的处理是由 make 进行的。

  示例 :

  文件： Makefile

  命令： make shellfun

$(warning TEXT...)

- 函数功能：

  提示“ TEXT...” 信息给用户，并不退出 make 的执行。

  返回值：空

  示例：

  文件： Makefile

  命令： make

自动化变量

- $@ 表示规则的目标文件名。
- $< 规则的第一个依赖文件名。
- $? 所有比目标文件更新的依赖文件列表，空格分割。
- $^ 规则的所有依赖文件列表，使用空格分隔。

示例：

文件： Makefile

命令： make autovar

谢谢大家！

- 联系方式
- 手机： 15010404682
- 邮箱： qin_fengling@126.com
- QQ： 415508683