Makefile语法

本文主要记录Makefile用到的一些语法

**1、赋值运算符=， :=， ?= ，+=。**

新建一个Makefile，内容为：  
ifdef DEFINE\_VRE

VRE = "Hello World\_DEFINE\_VAR\_TRUE\_symbol\_"=""

else

endif

ifeq ($(OPT),define)

VRE ?= "Hello World\_OPT\_define"

endif

ifeq ($(OPT),add)

VRE += "Hello World\_OPT\_add"

endif

ifeq ($(OPT),recover)

VRE := "Hello World\_OPT\_recover"

endif

all:

@echo $(VRE) #command need use tab key, is not space key

敲入以下make命令：  
johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE=TRUE OPT=define

Hello World\_DEFINE\_VAR\_TRUE\_symbol\_=

johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE=TRUE OPT=add

Hello World\_DEFINE\_VAR\_TRUE\_symbol\_= Hello World\_OPT\_add

johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE=TRUE OPT=recover

Hello World\_OPT\_recover

johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE= OPT=define

Hello World\_OPT\_define

johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE= OPT=add

Hello World\_OPT\_add

johnma@ntd-OptiPlex-9020:~/test$ make DEFINE\_VRE= OPT=recover

Hello World\_OPT\_recover

从上面的结果中我们可以清楚的看到他们的区别了  
= 是最基本的赋值  
:= 是覆盖之前的值  
?= 是如果没有被赋值过就赋予等号后面的值  
+= 是添加等号后面的值

之前一直纠结makefile中“=”和“:=”的区别到底有什么区别。

1、“=”

make会将整个makefile展开后，再决定变量的值。也就是说，变量的值将会是整个makefile中最后被指定的值。看例子：

x = foo  
y = $(x) bar  
x = xyz

在上例中，y的值将会是 xyz bar ，而不是 foo bar 。

2、“:=”

“:=”表示变量的值决定于它在makefile中的位置，而不是整个makefile展开后的最终值。

x := foo  
y := $(x) bar  
x := xyz

在上例中，y的值将会是 foo bar ，而不是 xyz bar 了。

**2、自动化变量符号$@，$^，$<，$?。**

新建一个Makefile，内容为：

main: main.o hello.o hi.o

gcc -o main main.o hello.o hi.o

main.o: main.c

cc -c main.c

hello.o: hello.c

cc -c hello.c

hi.o: hi.c

cc -c hi.c

clean:

rm \*.o

rm main

改为

main: main.o hello.o hi.o

gcc -o $@ $^

main.o: main.c

cc -c $<

hello.o: hello.c

cc -c $<

hi.o: hi.c

cc -c $<

clean:

rm \*.o

rm main

$@ 表示目标文件

$^ 表示所有的依赖文件

$< 表示第一个依赖文件

$? 表示比目标还要新的依赖文件列表