**4.1 显示命令**

通常，make会把其要执行的命令行在命令执行前输出到屏幕上。

当我们用“@”字符在命令行前，那么，这个命令将不被make显示出来

**4.2 命令执行**

比如你的第一条命令是cd命令，你希望第二条命令得在cd之后的基础上运行，那么你就不能把这两条命令写在两行上，而应该把这两条命令写在一行上，用分号分隔。如：

   示例二：

       exec:

               cd /home/hchen; pwd

当我们执行“make exec”时，pwd会打印出“/home/hchen”。

**4.3 命令出错**

有些时候，我们就不希望mkdir出错而终止规则的运行。

为了做到这一点，忽略命令的出错，我们可以在Makefile的命令行前加一个减号“-”（在Tab键之后）如：

  clean:

           -rm -f \*.o

给make加上“-i”或是“--ignore-errors”参数，那么，Makefile中所有命令都会忽略错误。

**4.4 嵌套执行make**

       在一些大的工程中，我们会把我们不同模块或是不同功能的源文件放在不同的目录中，在每个目录中都书写一个该目录的Makefile

      例如，我们有一个子目录叫subdir，这个目录下有个Makefile文件，来指明了这个目录下文件的编译规则。那么我们总控的Makefile可以这样书写：

   subsystem:

           cd subdir && $(MAKE)

其等价于：

    subsystem:

           $(MAKE) -C subdir

这两个例子的意思都是先进入“subdir”目录，然后执行make命令。

我们把这个Makefile叫做“总控Makefile”，总控Makefile的变量可以传递到下级的Makefile中（如果你显示的声明），但是不会覆盖下层的Makefile中所定义的变量，除非指定了“-e”参数。

如果你要传递变量到下级Makefile中，那么你可以使用这样的声明：

export<variable ...>

如果你不想让某些变量传递到下级Makefile中，那么你可以这样声明：

unexport<variable ...>

如：

      示例一：

        export variable = value

  其等价于：

        variable = value

        export variable

     如果你要传递所有的变量，那么，只要一个export就行了。后面什么也不用跟，表示传递所有的变量。

**4.5 定义命令包**

定义命令包语法以“define”开始，以“endef”结束，如：

   define run-yacc

   yacc $(firstword $^)

   mv y.tab.c $@

   endef

这里，“run-yacc”是这个命令包的名字。在“define”和“endef”中的两行就是命令序列。

把这个命令包放到一个示例中来看看吧。

   foo.c : foo.y

           $(run-yacc)

要使用这个命令包，我们就好像使用变量一样。在这个命令包的使用中，命令包“run-yacc”中的“$^”就是“foo.y”，“$@”就是“foo.c”