**一：批处理文件**

批处理文件是扩展名为·bat 或·cmd的文本文件，包含一条或多条命令，由DOS或Windows系统内嵌的命令解释器来解释运行。

批处理的本质，是一堆DOS命令按一定顺序排列而形成的集合。

**二：常用命令：**

1. **echo 和 @：**

@ #关闭单行回显

echo off #从下一行开始关闭回显

@echo off #从本行开始关闭回显，一般批处理第一行都是这个

echo on #从下一行开始打开回显

echo #显示当前是 echo off 状态还是 echo on 状态

echo. #输出一个”回车换行”，空白行

echo “字符串”或%var% #在命令行窗口显示该字符串或变量内容。

#(同echo, echo; echo+ echo[ echo] echo/ echo")

1. **errorlevel**

echo %errorlevel%

每个命令运行结束，可以用这个命令行格式查看返回码

默认值为0，一般命令执行出错会设 errorlevel 为1

**目录操作类命令：**

1. **dir 显示磁盘目录命令**

dir #显示当前目录中的文件和子目录

dir /a #显示当前目录中的文件和子目录，包括隐藏文件和系统文件

dir c: /a:d #显示 C 盘当前目录中的目录

dir c: /a:-d #显示 C 盘根目录中的文件

dir c:" /b/p #/b只显示文件名，/p分页显示

dir c:" /b/p #/w不显示详细信息

dir \*.exe /s #显示当前目录和子目录里所有的.exe文件

1. **以树形显示当前路径文件夹结构**

tree /f #显示每个文件夹中文件的名字。

**5.md 建立子目录**

md d:"a"b"c #如果 d:"a 不存在，将会自动创建中级目录

#如果命令扩展名被停用，则需要键入 mkdir "a"b"c

**6.cd 改变当前目录（进入）**

cd \ #进入根目录

cd #显示当前目录

cd /d d:\sdk #可以同时更改盘符和目录

cd .. #返回上一级目录

**7.rd 删除子目录命令**

rd abc #删除当前目录里的 abc 子目录，要求为空目录

rd /s/q d:\temp #删除 d:"temp 文件夹及其子文件夹和文件/q安静模式

**磁盘操作类命令：**

**8.format 磁盘格式化命令**

**9.chkdsk 检查磁盘当前状态命令**

**10.label 建立磁盘卷标命令**

**11.vol 显示磁盘卷标命令**

**文件操作类命令：**

1. **copy 文件复制命令**

copy c:\test.txt d:\test.bak

复制 c:\test.txt 文件到 d:\ ，并重命名为 test.bak

copy con test.txt

从屏幕上等待输入，按 Ctrl+Z 结束输入，输入内容存为test.txt文件

con代表屏幕，prn代表打印机，nul代表空设备

copy 1.txt + 2.txt 3.txt

合并 1.txt 和 2.txt 的内容，保存为 3.txt 文件

如果不指定 3.txt ，则保存到 1.txt

copy test.txt +

复制文件到自己，实际上是修改了文件日期

1. **xcopy 目录复制命令**

xcopy c:\ d:\

1. **ren 文件改名命令**

ren d:\temp tmp #支持对文件夹的重命名

1. **attrib 修改文件属性命令**

attrib filename #查看文件属性

+ #设置属性

- #清除属性

r #只读文件

a #存档文件

s #系统文件

H #隐藏文件

/S #处理当前文件夹及其所有子文件夹中的匹配文件

/D #处理文件夹

1. **type 显示文件内容**

type c:\boot.ini #显示指定文件的内容，程序文件一般会显示乱码

type \*.txt #显示当前目录里所有.txt文件的内容

1. **del 删除文件命令**

del d:\test.txt #删除指定文件，不能是隐藏、系统、只读文件

del /q/a/f d:"temp"\*.\* #删除 d:"temp 文件夹里面的所有文件包括

隐藏、只读、系统文件，不包括子目录

del /q/a/f/s d:"temp"\*.\* #删除 d:"temp 及子文件夹里面的所有文件包括

隐藏、只读、系统文件，不包括子目录

**其他命令：**

1. **CLS 清屏幕命令**
2. **ver 查看系统版本号命令**
3. **time,date 系统时钟日期设置/显示命令**

date #显示当前日期，并提示输入新日期，按""回车""略过输入

date/t #只显示当前日期，不提示输入新日期

time #显示当前时间，并提示输入新时间，按""回车""略过输入

time/t #只显示当前时间，不提示输入新时间

1. **rem 和 :: 注释**

注释行不执行操作

1. **color 设置默认的控制台前景和背景颜色**

**逻辑命令：**

1. **goto 和 :**

跳转命令

:label #行首为:表示该行是标签行，标签行不执行操作

goto label #跳转到指定的标签那一行

1. **set 定义变量**

引用变量可在变量名前后加 % ，即 %变量名%

set #显示目前所有可用的变量，包括系统变量和自定义的变量

echo %SystemDrive% #显示系统盘盘符。系统变量可以直接引用

set p #显示所有以p开头的变量，有就设errorlevel=1

set p=aa1bb1aa2bb2 #设置变量p，并赋值为 = 后面的字符串，即aa1bb1aa2bb2

echo %p% #显示变量p代表的字符串，即aa1bb1aa2bb2

echo %p:~6% #显示变量p中第6个字符以后的所有字符，即aa2bb2

echo %p:~6,3% #显示第6个字符以后的3个字符，即aa2

echo %p:~0,3% #显示前3个字符，即aa1

echo %p:~-2% #显示最后面的2个字符，即b2

echo %p:~0,-2% #显示除了最后2个字符以外的其它字符，即aa1bb1aa2b

echo %p:aa=c% #用c替换变量p中所有的aa，即显示c1bb1c2bb2

echo %p:aa=% #将变量p中的所有aa字符串置换为空，即显示1bb12bb2

echo %p:\*bb=c% #第一个bb及其之前的所有字符被替换为c，即显示c1aa2bb2

set p=%p:\*bb=c% #设置变量p，赋值为 %p:\*bb=c% ，即c1aa2bb2

set /a p=39 #设置p为数值型变量，值为39

set /a p=39/10 #支持运算符，有小数时用去尾法，去尾得3，p=3

set /a p=p/10 #用 /a 参数时，在 = 后面的变量可以不加%直接引用

set /a p=”1&0″ #”与”运算，要加引号。其它支持的运算符参见set/?

set p= #取消p变量

set /p p=请输入

屏幕上显示”请输入”，并会将输入的字符串赋值给变量p注意这条可以用来取代 choice 命令

注意变量在 if 和 for 的复合语句里是一次性全部替换的，如

@echo off

set p=aaa

if %p%==aaa (

echo %p%

set p=bbb

echo %p%

)

结果将显示

aaa

aaa

因为在读取 if 语句时已经将所有 %p% 替换为aaa

这里的""替换""，在 /? 帮助里就是指""扩充""、""环境变量扩充""

可以启用”延缓环境变量扩充”，用 ! 来引用变量，即 !变量名!

@echo off

SETLOCAL ENABLEDELAYEDEXPANSION

set p=aaa

if %p%==aaa (

echo %p%

set p=bbb

echo !p!

)

ENDLOCAL

结果将显示

aaa

bbb

还有几个动态变量，运行 set 看不到

%CD% #代表当前目录的字符串

%DATE% #当前日期

%TIME% #当前时间

%RANDOM% #随机整数，介于0~32767

%ERRORLEVEL% #当前 ERRORLEVEL 值

%CMDEXTVERSION% #当前命令处理器扩展名版本号

%CMDCMDLINE% #调用命令处理器的原始命令行

可以用echo命令查看每个变量值，如 echo %time%

注意 %time% 精确到毫秒，在批处理需要延时处理时可以用到

[变量大全]

系统变量:

系统将其根据事先定义的条件自动赋值,也就是系统已经给他们定义了值,

不需要我们来给他赋值,我们只需要调用。

%ALLUSERSPROFILE% 本地 返回“所有用户”配置文件的位置。

%APPDATA% 本地 返回默认情况下应用程序存储数据的位置。

%CD% 本地 返回当前目录字符串。

%CMDCMDLINE% 本地 返回用来启动当前的 Cmd.exe 的准确命令行。

%CMDEXTVERSION% 系统 返回当前的“命令处理程序扩展”的版本号。

%COMPUTERNAME% 系统 返回计算机的名称。

%COMSPEC% 系统 返回命令行解释器可执行程序的准确路径。

%DATE% 系统 返回当前日期。使用与 date /t 命令相同的格式。由 Cmd.exe 生成。

%ERRORLEVEL% 系统 返回上一条命令的错误代码。通常用非零值表示错误。

%HOMEDRIVE% 系统 返回连接到用户主目录的本地工作站驱动器号。基于主目录值而设置。用

户主目录是在“本地用户和组”中指定的。

%HOMEPATH% 系统 返回用户主目录的完整路径。基于主目录值而设置。用户主目录是在“本

地用户和组”中指定的。

%HOMESHARE% 系统 返回用户的共享主目录的网络路径。基于主目录值而设置。用户主目录是

在“本地用户和组”中指定的。

%LOGONSERVER% 本地 返回验证当前登录会话的域控制器的名称。

%NUMBER\_OF\_PROCESSORS% 系统 指定安装在计算机上的处理器的数目。

%OS% 系统 返回操作系统名称。Windows 2000 显示其操作系统为 Windows\_NT。

%PATH% 系统 指定可执行文件的搜索路径。

%PATHEXT% 系统 返回操作系统认为可执行的文件扩展名的列表。

%PROCESSOR\_ARCHITECTURE% 系统 返回处理器的芯片体系结构。

%PROCESSOR\_IDENTFIER% 系统 返回处理器说明。

%PROCESSOR\_LEVEL% 系统 返回计算机上安装的处理器的型号。

%PROCESSOR\_REVISION% 系统 返回处理器的版本号。

%PROMPT% 本地 返回当前解释程序的命令提示符设置。由 Cmd.exe 生成。

%RANDOM% 系统 返回 0 到 32767 之间的任意十进制数字。由 Cmd.exe 生成。

%SYSTEMDRIVE% 系统 返回包含 Windows server operating system 根目录（即系统根目录）

的驱动器。

%SYSTEMROOT% 系统 返回 Windows server operating system 根目录的位置。

%TEMP% 和 %TMP% 系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录。

%TIME% 系统 返回当前时间。使用与 time /t 命令相同的格式。由 Cmd.exe 生成。

%USERDOMAIN% 本地 返回包含用户帐户的域的名称。

%USERNAME% 本地 返回当前登录的用户的名称。

%USERPROFILE% 本地 返回当前用户的配置文件的位置。

%WINDIR% 系统 返回操作系统目录的位置。

1. **pause 暂停命令**
2. **if 判断命令**

if ERRORLEVEL number command #判断命令是否正确运行

通常情况下，命令执行成功后，系统变量ERRORLEVEL会被设置成0，

若命令失败，或遇到非法命令，系统变量ERRORLEVEL会被设置成非0。

if string1==string2 command #判断某两个字符串是否相等

if exit filename command #判断驱动器、文件或文件夹是否存在

if defined var command #判断某个变量是否已经被赋值

if 1 equ 2 #判断某两个数值是否相等

EQU #等于 ==

NEQ #不等于 ！=

LSS #小于 <

LEQ #小于或等于 <=

GTR #大于 >

GEQ #大于或等于 >=

注： #ELSE 子句必须出现在同一行上的 IF 之后

Example:

if ""%1""==""/a"" echo

第一个参数是/a

if /i ""%1"" equ ""/a"" echo

第一个参数是/a

/i 表示不区分大小写，equ 和 == 是一样的

其它运算符参见 if/?

if exist c:\test.bat echo 存在c:\test.bat文件

if not exist c:\windows (

echo 不存在c:"windows文件夹

)

if exist c:\test.bat (

echo 存在c:\test.bat

) else (

echo 不存在c:\test.bat

)

if exist 文件名　 待执行的命令

如果有指定的文件，则条件成立，运行命令，否则运行下一句。

如if exist config.sys edit config.sys

if errorlevel / if not errorlevel 数字　 待执行的命令

如果返回码等于指定的数字，则条件成立，运行命令，否则运行下一句。

如if errorlevel 2 goto x2

1. **call 命令 调用子程序**

CALL [drive:][path]filename [batch-parameters]

batch-parameters 指定批处理程序所需的命令行信息（参数）。

1. **exit命令 退出命令**
2. **for循环**

/F #分块提取文件

/D #显示当前目录下所有指定的目录名称

/R #枚举文件和目录

/L #计次循环

子参数：tokens（块）、delims（分块标志）、eol（忽略行）、skip（跳过行）、userbackq（支持长路径名)

CMD窗口中：

for %I in (command1) do command2

批处理中：

for %%I in (command1) do command2

1、for、in和do是for语句的关键字，它们三个缺一不可；

2、%%I是for语句中对形式变量的引用，就算它在do后的语句中没有参与语句的执行，也是必须出现的；

3、in之后，do之前的括号不能省略；

4、command1表示字符串或变量，command2表示字符串、变量或命令语句；

@echo off

for %%I in (bbs.jksing.com) do echo %%I

pause

1、for语句的形式变量I，可以换成26个字母中的任意一个，这些字母会区分大小写，也就是说，%%I和%%i会被认为不是同一个变量；形式变量I还可以换成其他的字符，但是，为了不与批处理中的%0～%9这10个形式变量发生冲突，请不要随意把%%I替换为%%0 ～%%9中的任意一个；

2、in和do之间的command1表示的字符串或变量可以是一个，也可以是多个，每一个字符串或变量，我们称之为一个元素，每个元素之间，用空格键、跳格键、逗号、分号或等号分隔；

3、for语句依次提取command1中的每一个元素，把它的值赋予形式变量I,带到do后的command2中参与命令的执行；并且每次只提取一个元素，然后执行一次do后的命令语句，而无论这个元素是否被带到command2中参与了command2的运行；当执行完一次do后的语句之后，再提取command1中的下一个元素，再执行一次command2，如此循环，直到command1中的所有元素都已经被提取完毕，该for语句才宣告执行结束；

Example:

for /f "tokens=1 delims= " %%a in (test.txt) do (

echo %%a

)

取第1行并显示

for /f "tokens=1,2 delims= " %%a in (test.txt) do (

echo %%a %%b

)

取1,2行并显示

for /f "tokens=1-3 delims= " %%a in (test.txt) do (

echo %%a %%b %%c

)

取1到3行并显示

for /f "tokens=2,3,\* delims= " %%a in (test.txt) do (

echo %%a %%b %%c

)

取2,3行并显示

for /f "eol=! tokens=\*" %%a in (test.txt) do (

echo %%a

)

忽略！开头的行并显示

for /f "skip=3 tokens=\*" %%a in (test.txt) do (

echo %%a

)

忽略前三行并显示

for /d %%a in (pro\*,doc\*) do (

echo %%a

)

显示以pro开头的目录名称，显示以doc开头的目录名称

for /r %systemroot% %%a in (\*.tmp) do (

echo %%a

)

显示root目录下后缀名为.tmp的文件及目录

set i=0

for /l %%a in (1,1,100) do (

set /a i+=1

)

echo 结算结果：%i%

(变量起始值，每次递加的步长，终止值)

for /f "usebackq" %%a in ("C:\Program Files\jksing.txt") do (

echo %%a

)

显示jksing中的内容，若去掉usebackq，则输出C:\Program

for /r %%i in (\*.txt) do (

echo %%~fi

)

for %%i in (c: d: e: f:) do echo %%i

依次调用小括号里的每个字符串，执行 do 后面的命令

注意%%i，在批处理中 for 语句调用参数用2个%

默认的字符串分隔符是""空格键""，""Tab键""，""回车键""

for %%i in (\*.txt) do find ""abc"" %%i

对当前目录里所有的txt文件执行 find 命令

for /r . %%i in (\*.txt) do find ""abc"" %%i

在当前目录和子目录里所有的.txt文件中搜索包含 abc 字符串的行

for /r . %%i in (.) do echo %%~pn

显示当前目录名和所有子目录名，包括路径，不包括盘符

for /r d:"mp3 %%i in (\*.mp3) do echo %%i>>d:"mp3.txt

把 d:"mp3 及其子目录里的mp3文件的文件名都存到 d:"mp3.txt 里去

for /l %%i in (2,1,8) do echo %%i

生成2345678的一串数字，2是数字序列的开头，8是结尾，1表示每次加1

for /f %%i in ('set') do echo %%i

对 set 命令的输出结果循环调用，每行一个

for /f ""eol=P"" %%i in ('set') do echo %%i

取 set 命令的输出结果，忽略以 P 开头的那几行

for /f %%i in (d:"mp3.txt) do echo %%i

显示 d:"mp3.txt 里的每个文件名，每行一个，不支持带空格的名称

for /f ""delims="" %%i in (d:"mp3.txt) do echo %%i

显示 d:"mp3.txt 里的每个文件名，每行一个，支持带空格的名称

for /f ""skip=5 tokens=4"" %%a in ('dir') do echo %%a

对 dir 命令的结果，跳过前面5行，余下的每行取第4列

每列之间的分隔符为默认的""空格""

可以注意到 dir 命令输出的前5行是没有文件名的

for /f ""tokens=1,2,3 delims=- "" %%a in ('date /t') do (

echo %%a

echo %%b

echo %%c

)

对 date /t 的输出结果，每行取1、2、3列

第一列对应指定的 %%a ，后面的 %%b 和 %%c 是派生出来的，对应其它列

分隔符指定为 - 和""空格""，注意 delims=- 后面有个""空格""

其中 tokens=1,2,3 若用 tokens=1-3 替换，效果是一样的

for /f ""tokens=2\* delims=- "" %%a in ('date /t') do echo %%b

取第2列给 %%a ，其后的列都给 %%b

1. **参数%0 %1 %2 %3 %4 %5 %6 %7 %8 %9 %\* i%%**

命令行传递给批处理的参数

%0 #批处理文件本身

%1 #第一个参数

%9 #第九个参数

%\* #从第一个参数开始的所有参数

参数扩展：

%~1 #删除引号("")，扩充 %1

%~f1 #将 %1 扩充到一个完全合格的路径名

%~d1 #仅将 %1 扩充到一个驱动器号

%~p1 #仅将 %1 扩充到一个路径

%~n1 #仅将 %1 扩充到一个文件名

%~x1 #仅将 %1 扩充到一个文件扩展名

%~s1 #扩充的路径指含有短名

%~a1 #将 %1 扩充到文件属性

%~t1 #将 %1 扩充到文件的日期/时间

%~z1 #将 %1 扩充到文件的大小

%~$PATH #查找列在 PATH 环境变量的目录，并将 %1扩充到找到的第一个完全合格的名称。如果环境变量名未被定义，或者没有找到文件，此组合键会扩充到空字符串

可以组合修定符来取得多重结果:

%~dp1 #只将 %1 扩展到驱动器号和路径

%~nx1 #只将 %1 扩展到文件名和扩展名

%~dp$PATH: #在列在 PATH 环境变量中的目录里查找 %1，并扩展到找到的第一个文件的驱动器号和路径

%~ftza1 #将 %1 扩展到类似 DIR 的输出行

1. **find 查找命令(外部命令)**

在文件中搜索字符串

find ""abc"" c:"test.txt

在 c:"test.txt 文件里查找含 abc 字符串的行

如果找不到，将设 errorlevel 返回码为1

find /i “abc” c:"test.txt

查找含 abc 的行，忽略大小写

find /c ""abc"" c:"test.txt

显示含 abc 的行的行数

1. **more 逐屏显示输出**

more c:"test.txt #逐屏显示 c:"test.txt 的文件内容

1. **assoc 和 ftype**

文件关联

assoc 设置'文件扩展名'关联，关联到'文件类型'

ftype 设置'文件类型'关联，关联到'执行程序和参数'

当你双击一个.txt文件时，windows并不是根据.txt直接判断用 notepad.exe 打开

而是先判断.txt属于 txtfile '文件类型'

再调用 txtfile 关联的命令行 txtfile=%SystemRoot%"system32"NOTEPAD.EXE %1

可以在""文件夹选项""→""文件类型""里修改这2种关联

assoc #显示所有'文件扩展名'关联

assoc .txt #显示.txt代表的'文件类型'，结果显示 .txt=txtfile

assoc .doc #显示.doc代表的'文件类型'，结果显示 .doc=Word.Document.8

assoc .exe #显示.exe代表的'文件类型'，结果显示 .exe=exefile

ftype #显示所有'文件类型'关联

ftype exefile #显示exefile类型关联的命令行，结果显示 exefile=""%1"" %\*

assoc .txt=Word.Document.8

设置.txt为word类型的文档，可以看到.txt文件的图标都变了

assoc .txt=txtfile

恢复.txt的正确关联

ftype exefile=""%1"" %\*

恢复 exefile 的正确关联

如果该关联已经被破坏，可以运行 command.com ，再输入这条命令

1. **pushd 和 popd**

切换当前目录

@echo off

c: & cd" & md mp3 #在 C:" 建立 mp3 文件夹

md d:"mp4 #在 D:" 建立 mp4 文件夹

cd /d d:"mp4 #更改当前目录为 d:"mp4

pushd c:"mp3 #保存当前目录，并切换当前目录为 c:"mp3

popd #恢复当前目录为刚才保存的 d:"mp4

1. **subst 映射磁盘**

subst z: "server"d #这样输入z:就可以访问"server"d了

subst z: /d #取消该映射

subst #显示目前所有的映时

**常用特殊符号:**

1.@ #命令行回显屏蔽符

2.% #批处理变量引导符

3.> #重定向符

4.>> #重定向符

5.<,>&,<& #重定向符

6.| #命令管道符

7.^ #转义字符

8.& #组合命令

9.&& #组合命令

10.|| #组合命令

11."" #字符串界定符

**详细介绍：**

1. **% 批处理变量引导符**
2. **> ，>> 重定向符**

> 清除文件中原有的内容后再写入

>> 追加内容到文件末尾，而不会清除原有的内容主要将本来显示在屏幕上的内容输出到指定文件中

指定文件如果不存在，则自动生成该文件

Example：

echo hello > 1.txt

echo world >>1.txt

这时候 1.txt 内容如下:

hello

world

type c:"test.txt >prn

屏幕上不显示文件内容，转向输出到打印机

echo hello world>con

在屏幕上显示hello world，实际上所有输出都是默认 >con 的

copy c:"test.txt f: >nul

拷贝文件，并且不显示""文件复制成功""的提示信息，但如果f盘不存在，还是会显示出错信息

copy c:"test.txt f: >nul 2>nul

不显示”文件复制成功”的提示信息，并且f盘不存在的话，也不显示错误提示信息

echo ^^W ^> ^W>c:"test.txt

生成的文件内容为 ^W > W

^ 和 > 是控制命令，要把它们输出到文件，必须在前面加个 ^ 符号

1. **<，>&，<& 重定向符**

< 从文件中获得输入信息,而不是从屏幕上,一般用date time label等需要等待输入的命令

>& 将一个句柄的输出写入到另一个句柄的输入中

<& 刚好和>&相反，从一个句柄读取输入并将其写入到另一个句柄输出中

Example:

@echo off

echo 2005-05-01>temp.txt

date <temp.txt

del temp.txt

这样就可以不等待输入直接修改当前日期

常用句柄：0、1、2，未定义句柄：3—9

1>nul 表示禁止输出正确的信息

2>nul 表示禁止输出错误信息。

其中的 1 与 2 都是代表某个数据流输入输出的地址（NT CMD 称之为句柄，MSDOS 称之为设备）。

句柄 0 ：标准输入stdin，键盘输入

句柄 1：标准输出stdout，输出到命令提示符窗口（console，代码为 CON ）

句柄 2：标准错误stderr，输出到命令提示符窗口（console，代码为 CON ）

其中的 stdin 可被<重定向，stdout 可被>、>>重定向。

我们已经知道读取文本中的内容可以用 for 命令，但如果只需要读取第一行用 for 命令就有点麻烦。简单的

办法如下:

@echo off

set /p str=<%0

echo %str%

pause

运行显示批处理文件自身的第一行：@echo off

1. **& ,&& ,|| 组合命令**

& 、&& 、||为组合命令，顾名思义，就是可以把多个命令组合起来当一个命令来执行。

这在批处理脚本里 是允许的，而且用的非常广泛。因为批处理认行不认命令数目。

**&:**

语法：第一条命令 & 第二条命令 [& 第三条命令...]

&允许在一行中使用 2 个以上不同的命令，当第一个命令执行失败了，也不影响后边的命令执行。

这里&两边的命令是顺序执行的，从前往后执行。

比如：

dir z:\ & dir y:\ & dir c:\

以上命令会连续显示z,y,c 盘的内容，不理会该盘是否存在

**&&:**

语法：第一条命令 && 第二条命令 [&& 第三条命令...]

用这种方法可以同时执行多条命令，当碰到执行出错的命令后将不执行后面的命令，如果一直没有出错则 一直执行完所有命令

这个命令和上边的类似，但区别是，第一个命令失败时，后边的命令也不会执行

dir z:\ && dir y:\ && dir c:\

**||:**

语法：第一条命令 || 第二条命令 [|| 第三条命令...]

用这种方法可以同时执行多条命令，当一条命令失败后才执行第二条命令，当碰到执行正确的命令后将不

执行后面的命令，如果没有出现正确的命令则一直执行完所有命令；

提示：组合命令和重定向命令一起使用必须注意优先级

管道命令的优先级高于重定向命令，重定向命令的优先级高于组合命令

1. **| 命令管道符**

格式：第一条命令 | 第二条命令 [| 第三条命令...]

将第一条命令的结果作为第二条命令的参数来使用，记得在 unix 中这种方式很常见。

例如：

dir \*.\* /s/a | find /c "".exe""

管道命令表示先执行 dir 命令，对其输出的结果执行后面的 find 命令

该命令行结果：输出当前文件夹及所有子文件夹里的.exe文件的个数

type c:"test.txt|more

这个和 more c:"test.txt 的效果是一样的

dir c:\|find "txt"

以上命令是：查找C：\所有，并发现 TXT 字符串。

FIND 的功能请用 FIND /? 自行查看

在不使 format 的自动格式化参数时，我是这样来自动格式化A 盘的

echo y|format a: /s /q /v:system

用过 format 的都知道，再格盘时要输入y 来确认是否格盘，这个命令前加上 echo y 并用|字符来将 echo y

的结果传给format 命令

从而达到自动输入 y 的目的

1. **^ 转义字符**

^是对特殊符号<,>,& 的前导字符，在命令中他将以上 3 个符号的特殊功能

去掉仅只把他们当成符号而不使用他们的特殊意义。

Example:

echo test ^>1.txt

结果则是：test > 1.txt

他没有追加在 1.txt 里，只是显示了出来

另外，此转义字符还可以用作续行符号。

@echo off

echo 英雄^

是^

好^

男人

pause

为什么转义字符放在行尾可以起到续行符的作用呢？原因很简单，因为每行末尾还有一个看不见的符号,即回车符，转义字符位于行尾时就让回车符失效了，从而起到了续行的作用

1. **"" 字符串界定符**

双引号允许在字符串中包含空格，进入一个特殊目录可以用如下方法

cd "program files"

cd progra~1

cd pro\*

以上三种方法都可以进入 program files 这个目录

1. **; 分号**

分号，当命令相同时，可以将不同目标用；来隔离

但执行效果不变，如执行过程中发生错误则只返回错误报告，但程序仍会执行。

（有人说不会继续执行，其实测试一下就知道了）

比如：

dir c:\;d:\;e:\;z:\

以上命令相当于

dir c:\

dir d:\

dir e:\

dir f:\

如果其中 z 盘不存在，运行显示：系统找不到指定的路径。然后终止命令的执行。

例：dir c:\;d:\;e:\1.txt

以上命令相当于

dir c:\

dir d:\

dir e:\1.txt

其中文件 e:\1.txt 不存在，但 e 盘存在，有错误提示，但命令仍会执行。

如果目标路径不存在，则终止执行；如果路径存在，仅文件不存在，则继续执行。

1. **() 括号**

小括号在批处理编程中有特殊的作用，左右括号必须成对使用

括号中可以包括多行命令，这些命令将被看成一个整体，视为一条命令行

括号在 for 语句和 if 语句中常见，用来嵌套使用循环或条件语句，其实括号()也可以单独使用，请看例子。

Example:

echo 1 & echo 2 & echo 3

可以写成：

(

echo 1

echo 2

echo 3

)

上面两种写法效果一样，这两种写法都被视为是一条命令行。

注意：这种多条命令被视为一条命令行时，如果其中有变量，就涉及到变量延迟的问题

1. **! 感叹号**

在变量延迟问题中，用来表示变量，即%var%应该表示为!var!

**DOS 编程高级技巧 :**

一、交互界面设计

没啥说的，看看高手设计的菜单界面吧：

@echo off

cls

title 终极多功能修复

:menu

cls

color 0A

echo.

echo ==============================

echo 请选择要进行的操作，然后按回车

echo ==============================

echo.

echo 1. 网络修复及上网相关设置,修复IE, 自定义屏蔽网站

echo.

echo 2. 病毒专杀工具，端口关闭工具,关闭自动播放

echo.

echo 3. 清除所有多余的自启动项目，修复系统错误

echo.

echo 4. 清理系统垃圾,提高启动速度

echo.

echo Q.退出

echo.

echo.

:cho

set choice=

set /p choice= 请选择 :

IF NOT "%choice%"=="" SET choice=%choice:~0,1%

if /i "%choice%"=="1" goto ip

if /i "%choice%"=="2" goto setsave

if /i "%choice%"=="3" goto kaiji

if /i "%choice%"=="4" goto clean

if /i "%choice%"=="Q" goto endd

echo 选择无效，请重新输入

echo.

goto cho

只要学完本教程前面的章节，上面的程序应该能看懂了。

二、if …else …条件语句

前面已经谈到，DOS 条件语句主要有以下形式

IF [NOT] ERRORLEVEL number command

IF [NOT] string1==string2 command

IF [NOT] EXIST filename command

增强用法：IF string1 compare-op string2 command

增强用法中加上/I 就不区分大小写了!

增强用法中还有一些用来判断数字的符号：

EQU - 等于

NEQ - 不等于

LSS - 小于

LEQ - 小于或等于

GTR - 大于

GEQ - 大于或等于

上面的command命令都可以用小括号来使用多条命令的组合，包括 else 子句，组合命令中可以嵌套使用条

件或循环命令。

例如 :

IF EXIST filename (

del filename

) ELSE (

echo filename missing

)

也可写成：

if exist filename (del filename) else (echo filename missing)

但这种写法不适合命令太多或嵌套命令的使用。

三、循环语句

1、指定次数循环

FOR /L %variable IN (start,step,end) DO command [command-parameters]

组合命令：

FOR /L %variable IN (start,step,end) DO (

Command1

Command2

……

)

2、对某集合执行循环语句。

FOR %%variable IN (set) DO command [command-parameters]

%%variable 指定一个单一字母可替换的参数。

(set) 指定一个或一组文件。可以使用通配符。

command 对每个文件执行的命令，可用小括号使用多条命令组合。

FOR /R [[drive:]path] %variable IN (set) DO command [command-parameters]

检查以 [drive:]path 为根的目录树，指向每个目录中的

FOR 语句。如果在 /R 后没有指定目录，则使用当前

目录。如果集仅为一个单点(.) 字符，则枚举该目录树。

同前面一样，command可以用括号来组合：

FOR /R [[drive:]path] %variable IN (set) DO (

Command1

Command2

……

commandn

)

3、条件循环

上面的循环结构是用 for 命令来实现的，for 命令循环有一个缺点，就是整个循环被当作一条命令语句，

涉及到变量延迟的问题。

利用 goto 语句和条件判断，dos 可以实现条件循环，很简单啦，看例子：

例：

@echo off

set var=0

rem \*\*\*\*\*\*\*\*\*\*\*\* 循环开始了

:continue

set /a var+=1

echo 第%var%次循环

if %var% lss 100 goto continue

rem \*\*\*\*\*\*\*\*\*\*\*\* 循环结束了

echo 循环执行完毕

pause

例：

@echo off

set var=100

rem \*\*\*\*\*\*\*\*\*\*\*\* 循环开始了

:continue

echo 第%var%次循环

set /a var-=1

if %var% gtr 0 goto continue

rem \*\*\*\*\*\*\*\*\*\*\*\* 循环结束了

echo 循环执行完毕

pause

四、子程序

在批处理程序中可以调用外部可运行程序，比如 exe 程序，也可调用其他批处理程序，这些也可以看作子

程序，但是不够方便，如果被调用的程序很多，就显得不够简明了，很繁琐。

在windowsXP 中，批处理可以调用本程序中的一个程序段，相当于子程序，这些子程序一般放在主程序

后面。

子程序调用格式：

CALL :label arguments

子程序语法：

:label

command1

command2

......

commandn

goto :eof

在子程序段中，参数%0指标签 :label

子过程一般放在最后，并且注意在主程序最后要加上 exit 或跳转语句，避免错误的进入子过程。

子程序和主程序中的变量都是全局变量，其作用范围都是整个批处理程序。

传至子程序的参数在 call 语句中指定，在子程序中用%1、%2至%9的形式调用，而子程序返回主程序的数

据只需在调用结束后直接引用就可以了，当然也可以指定返回变量，请看下面的例子。

子程序例 1：

@echo off

call :sub return 你好

echo 子程序返回值： %return%

pause

:sub

set %1=%2

goto :eof

运行结果：你好

子程序例 2：设计一个求多个整数相加的子程序

@echo off

set sum=0

call :sub sum 10 20 35

echo 数据求和结果： %sum%

pause

:sub

rem 参数1 为返回变量名称

set /a %1=%1+%2

shift /2

if not "%2"=="" goto sub

goto :eof

运行结果：65

在win98 系统中，不支持上面这种标号调用，须将子程序单独保存为一个批处理程序，然后调用。

五、用ftp 命令实现自动下载

ftp 是常用的下载工具，ftp 界面中有 40多个常用命令，自己学习了，不介绍了。这里介绍如何用 dos 命令

行调用ftp 命令，实现 ftp 自动登录，并上传下载，并自动退出 ftp 程序。

其实可以将 ftp 命令组合保存为一个文本文件，然后用以下命令调用即可。

ftp -n -s:[[drive:]path]filename

上面的filename 为ftp 命令文件，包括登录 IP 地址，用户名、密码、操作命令等

例：

open 90.52.8.3 ＃打开 ip

user iware ＃用户为 iware

password8848 ＃密码

bin ＃二进制传输模式

prompt

cd tmp1 ＃切换至 iware 用户下的 tmp1 目录

pwd

lcd d:\download ＃本地目录

mget \* ＃下载 tmp1 目录下的所有文件

bye ＃退出 ftp

六、用7-ZIP 实现命令行压缩和解压功能

语法格式：（详细情况见 7-zip 帮助文件，看得头晕可以跳过，用到再学）

7z <command> [<switch>...] <base\_archive\_name> [<arguments>...]

7z.exe 的每个命令都有不同的参数<switch>, 请看帮助文件

<base\_archive\_name>为压缩包名称

<arguments>为文件名称，支持通配符或文件列表

其中，7z 是至命令行压缩解压程序 7z.exe ，<command>是7z.exe 包含的命令，列举如下：

a ： Adds files to archive. 添加至压缩包

a 命令可用参数：

-i (Include)

-m (Method)

-p (Set Password)

-r (Recurse)

-sfx (create SFX)

-si (use StdIn)

-so (use StdOut)

-ssw (Compress shared files)

-t (Type of archive)

-u (Update)

-v (Volumes)

-w (Working Dir)

-x (Exclude)

b： Benchmark

d： Deletes files from archive. 从压缩包中删除文件

d 命令可用参数：

-i (Include)

-m (Method)

-p (Set Password)

-r (Recurse)

-u (Update)

-w (Working Dir)

-x (Exclude)

e ： Extract 解压文件至当前目录或指定目录

e 命令可用参数：

-ai (Include archives)

-an (Disable parsing of archive\_name)

-ao (Overwrite mode)

-ax (Exclude archives)

-i (Include)

-o (Set Output Directory)

-p (Set Password)

-r (Recurse)

-so (use StdOut)

-x (Exclude)

-y (Assume Yes on all queries)

l： Lists contents of archive.

t： Test

u： Update

x： eXtract with full paths 用文件的完整路径解压至当前目录或指定目录

x 命令可用参数：

-ai (Include archives)

-an (Disable parsing of archive\_name)

-ao (Overwrite mode)

-ax (Exclude archives)

-i (Include)

-o (Set Output Directory)

-p (Set Password)

-r (Recurse)

-so (use StdOut)

-x (Exclude)

-y (Assume Yes on all queries)

七、调用VBScript程序

使用 Windows 脚本宿主，可以在命令提示符下运行脚本。CScript.exe 提供了用于设置脚本属性的命令行开

关。

用法：CScript 脚本名称 [ 脚本选项...] [ 脚本参数...]

选项：

//B 批模式：不显示脚本错误及提示信息

//D 启用 Active Debugging

//E:engine 使用执行脚本的引擎

//H:CScript 将默认的脚本宿主改为 CScript.exe

//H:WScript 将默认的脚本宿主改为 WScript.exe （默认）

//I 交互模式（默认，与 //B 相对)

//Job:xxxx 执行一个 WSF 工作

//Logo 显示徽标（默认）

//Nologo 不显示徽标：执行时不显示标志

//S 为该用户保存当前命令行选项

//T:nn 超时设定秒：允许脚本运行的最长时间

//X 在调试器中执行脚本

//U 用 Unicode 表示来自控制台的重定向 I/O

“脚本名称”是带有扩展名和必需的路径信息的脚本文件名称，如 d:\admin\vbscripts\chart.vbs 。

“脚本选项和参数”将传递给脚本。脚本参数前面有一个斜杠 (/)。每个参数都是可选的；但不能在未指定

脚本名称的情况下指定脚本选项。如果未指定参数，则 CScript 将显示 CScript 语法和有效的宿主参数。

八、将批处理转化为可执行文件：

由于批处理文件是一种文本文件，任何人都可以对其进行随便编辑，不小心就会把里面的命令破坏掉，所

以如果将其转换成.com 格式的可执行文件，不仅执行效率会大大提高，而且不会破坏原来的功能，更能将

优先级提到最高。Bat2Com 就可以完成这个转换工作。

小知识：在 DOS 环境下，可执行文件的优先级由高到低依次为.com>.exe>.bat>.cmd ，即如果在同一目录下

存在文件名相同的这四类文件，当只键入文件名时，DOS 执行的是 name.com，如果需要执行其他三个文

件，则必须指定文件的全名，如 name.bat 。

这是一个只有 5.43K 大小的免费绿色工具，可以运行在纯 DOS 或DOS 窗口的命令行中，用法：Bat2Com

FileName ，这样就会在同一目录下生成一个名为 FileNme.com 的可执行文件，执行的效果和原来的.bat 文件

一样。

九、时间延迟

本条参考引用[英雄] 教程

什么是时间延迟？顾名思义，就是执行一条命令后延迟一段时间再进行下一条命令。

延迟的应用见下节：“模拟进度条”。

1、利用ping命令延时

例：

@echo off

echo 延时前：%time%

ping /n 3 127.0.0.1 >nul

echo 延时后：%time%

pause

解说：用到了 ping 命令的“/n ”参数，表示要发送多少次请求到指定的 ip 。本例中要发送 3 次请求到本机

的ip （127.0.0.1 ）。 127.0.0.1 可简写为 127.1。“>nul ”就是屏蔽掉 ping 命令所显示的内容。

2、利用for 命令延时

例：

@echo off

echo 延时前：%time%

for /l %%i in (1,1,5000) do echo %%i>nul

echo 延时后：%time%

pause

解说：原理很简单，就是利用一个计次循环并屏蔽它所显示的内容来达到延时的目的。

3、利用vbs 延迟函数，精确度毫秒，误差1000 毫秒内

例：

@echo off

echo %time%

call :delay 5000

echo %time%

pause

exit

:delay

echo WScript.Sleep %1>delay.vbs

CScript //B delay.vbs

del delay.vbs

goto :eof

运行显示：

10:44:06.45

10:44:11.95

请按任意键继续. . .

上面的运行结果显示实际延时了 5500毫秒，多出来的 500 毫秒时建立和删除临时文件所耗费的时间。误差

在一秒之内。

4、仅用批处理命令实现任意时间延迟，精确度10毫秒，误差50毫秒内

仅用批处理命令就可以实现延迟操作。

例：

@echo off

set /p delay=请输入需延迟的毫秒数：

set TotalTime=0

set NowTime=%time%

:: 读取起始时间，时间格式为：13:01:05.95

echo 程序开始时间： %NowTime%

:delay\_continue

set /a minute1=1%NowTime:~3,2%-100

:: 读取起始时间的分钟数

set /a second1=1%NowTime:~-5,2%%NowTime:~-2%0-100000

:: 将起始时间的秒数转为毫秒

set NowTime=%time%

set /a minute2=1%NowTime:~3,2%-100

:: 读取现在时间的分钟数

set /a second2=1%NowTime:~-5,2%%NowTime:~-2%0-100000

:: 将现在时间的秒数转为毫秒

set /a TotalTime+=(%minute2%-%minute1%+60)%%60\*60000+%second2%-%second1%

if %TotalTime% lss %delay% goto delay\_continue

echo 程序结束时间： %time%

echo 设定延迟时间： %delay%毫秒

echo 实际延迟时间： %TotalTime%毫秒

pause

运行显示：

请输入需延迟的毫秒数：6000

程序开始时间：15:32:16.37

程序结束时间：15:32:22.37

设定延迟时间：6000毫秒

实际延迟时间：6000毫秒

请按任意键继续. . .

实现原理：首先设定要延迟的毫秒数，然后用循环累加时间，直到累加时间大于等于延迟时间。

误差：windows 系统时间只能精确到 10毫秒，所以理论上有可能存在 10毫秒误差。

经测试，当延迟时间大于 500 毫秒时，上面的延迟程序一般不存在误差。当延迟时间小于 500 毫秒时，

可能有几十毫秒误差，为什么？因为延迟程序本身也是有运行时间的，同时系统时间只能精确到 10毫秒。

为了方便引用，可将上面的例子改为子程序调用形式：

@echo off

echo 程序开始时间： %Time%

call :delay 10

echo 实际延迟时间： %totaltime%毫秒

echo 程序结束时间： %time%

pause

exit

::----------- 以下为延时子程序--------------------

:delay

@echo off

if "%1"=="" goto :eof

set DelayTime=%1

set TotalTime=0

set NowTime=%time%

:: 读取起始时间，时间格式为：13:01:05.95

:delay\_continue

set /a minute1=1%NowTime:~3,2%-100

set /a second1=1%NowTime:~-5,2%%NowTime:~-2%0-100000

set NowTime=%time%

set /a minute2=1%NowTime:~3,2%-100

set /a second2=1%NowTime:~-5,2%%NowTime:~-2%0-100000

set /a TotalTime+=(%minute2%-%minute1%+60)%%60\*60000+%second2%-%second1%

if %TotalTime% lss %DelayTime% goto delay\_continue

goto :eof

十、模拟进度条

下面给出一个模拟进度条的程序。如果将它运用在你自己的程序中，可以使你的程序更漂亮。

@echo off

mode con cols=113 lines=15 &color 9f

cls

echo.

echo 程序正在初始化. . .

echo.

echo ┌──────────────────────────────────────┐

set/p= ■<nul

for /L %%i in (1 1 38) do set /p a= ■<nul&ping /n 1 127.0.0.1>nul

echo 100%%

echo └──────────────────────────────────────┘

pause

解说：“ set /p a= ■<nul ”的意思是：只显示提示信息“■”且不换行，也不需手工输入任何信息，这样可

以使每个“■”在同一行逐个输出。“ping /n 0 127.1>nul ”是输出每个“■”的时间间隔，即每隔多少时

间输出一个“■”。

十一、特殊字符的输入及应用

开始 -> 运行 -> 输入cmd -> edit -> ctrl+p（意思是允许输入特殊字符）-> 按ctrl+a 将会显示笑脸图案。

（如果要继续输入特殊字符请再次按 ctrl+p，然后ctrl+ 某个字母）

以上是特殊字符的输入方法，选自[ 英雄 ]教程，很管用的。也就是用编辑程序 edit 输入特殊字符，然后保存

为一文本文件，再在 windows 下打开此文件，复制其中的特殊符号即可。

一些简单的特殊符号可以在 dos 命令窗口直接输入，并用重定向保存为文本文件。

例：

C:>ECHO ^G>temp.txt

“^G”是用 Ctrl＋G 或Alt＋007 输入，输入多个^G可以产生多声鸣响。

特殊字符的应用也很有意思，这里仅举一例：退格键

退格键表示删除左边的字符，此键不能在文档中正常输入，但可以通过 edit 编辑程序录入并复制出来。即

“”。

利用退格键，可以设计闪烁文字效果

例：文字闪烁

@echo off

:start

set/p= 床前明月光<nul

:: 显示文字，光标停于行尾

ping -n 0 127.0.0.1>nul

:: 设置延迟时间

set /p a= <nul

:: 输出一些退格符将光标置于该行的最左端（退格符的数量可以自己调整）。

set /p a= <nul

:: 输出空格将之前输出的文字覆盖掉。

set /p a= <nul

:: 再次输出退格符将光标置于该行的最左端，这里的退格符数量一定不能比前面的空格数少。

:: 否则光标不能退到最左端。

goto start

例：输出唐诗一首，每行闪动多次

@echo off

setlocal enabledelayedexpansion

set str=床前明月光 疑是地上霜 举头望明月 低头思故乡

:: 定义字符串 str

for %%i in (%str%) do (

rem 由于str中含有空格，则以空格为分隔符将 str中的每一个部分依次赋给变量%%i。

set char=%%i

echo.

echo.

for /l %%j in (0,1,5) do (

set/p=!char:~%%j,1!<nul

rem 依次取出变量 char 中的每一个字符，并显示。

ping -n 0 127.0.0.1>nul

rem 设置输出每个字符的时间延迟。

)

call :hero %%i

)

pause>nul

exit

:hero

for /l %%k in (1,1,10) do (

ping /n 0 127.0.0.1>nul

set /p a= <nul

set /p a= <nul

set /p a= <nul

ping /n 0 127.0.0.1>nul

set /p a=%1<nul

)

:: 文字闪动

goto :eof

十二、随机数（%random%）的应用技巧

%RANDOM% 系统变量 返回 0 到 32767 之间的任意十进制数字。由 Cmd.exe 生成。

2 的15次方等于 32768 ，上面的0～32767 实际就是 15位二进制数的范围。

那么，如何获取 100 以内的随机数呢？很简单，将%RANDOM%按100 进行求余运算即可，见例子。

例：生成 5 个100 以内的随机数

@echo off

setlocal enabledelayedexpansion

for /L %%i in (1 1 5) do (

set /a randomNum=!random!%%100

echo 随机数：!randomNum!

)

pause

运行结果：（每次运行不一样）

随机数： 91

随机数： 67

随机数： 58

随机数： 26

随机数： 20

请按任意键继续. . .

求余数运算 set /a randomNum=!random!%%100 中的 100 可以是 1～32768 之间的任意整数。

总结：利用系统变量%random%，求余数运算%%，字符串处理等，可以实现很多随机处理。

思考题目：生成给定位数的随机密码

解答思路：将 26个英文字母或 10数字以及其它特殊字符组成一个字符串，随机抽取其中的若干字符。

参考答案 1：（简单）

@echo off

call :randomPassword 5 pass1 pass2

echo %pass1% %pass2%

pause

exit

:randomPassword

::--------- 生成随机密码

::---------%1 为密码长度，%2及以后为返回变量名称

::---------for 命令最多只能区分 31个字段

@echo off

set password\_len=%1

if not defined password\_len goto :eof

if %password\_len% lss 1 goto :eof

set wordset=a b c d e f g h i j k l m n o p q r s t u v w x y z

set return=

set num=0

:randomPassword1

set /a num+=1

set /a numof=%random%%%26+1

for /f "tokens=%numof% delims= " %%i in ("%wordset%") do set return=%return%%%i

if %num% lss %password\_le n% goto randomPassword1

if not "%2"=="" set %2=%return%

shift /2

if not "%2"=="" goto randomPassword

goto :eof

参考答案 2：（最优）

@echo off

call :randomPassword 6 pass1 pass2 pass3

echo %pass1% %pass2% %pass3%

pause

exit

:randomPassword

::--------- 生成随机密码

::---------%1 为密码长度，%2及以后为返回变量名称

::---------goto 循环、变量嵌套、命令嵌套

@echo off

if "%1"=="" goto :eof

if %1 lss 1 goto :eof

set password\_len=%1

set return=

set wordset=abcdefghijklmnopqrstuvwxyz023456789\_

::--------------------------- 循环

:randomPassword1

set /a numof=%random%%%36

call set return=%return%%%wordset:~%numof%,1%%

set /a password\_len-=1

if %password\_len% gtr 0 goto randomPassword1

::--------------------------- 循环

if not "%2"=="" set %2=%return%

shift /2

if not "%2"=="" goto randomPassword

goto :eof

说明：本例涉及到变量嵌套和命令嵌套的应用，见后。

十三、变量嵌套 与 命令嵌套

和其它编程语言相比，dos 功能显得相对简单，要实现比较复杂的功能，需要充分运用各种技巧，变量嵌

套与命令嵌套就是此类技巧之一。

先复习一下前面的“字符串截取”的关键内容：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

截取功能统一语法格式为：%a:~[m[,n]]%

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

方括号表示可选，%为变量标识符，a 为变量名，不可少，冒号用于分隔变量名和说明部分，符号～可以简

单理解为“偏移”即可，m 为偏移量（缺省为 0），n 为截取长度（缺省为全部）。

百分号如果需要当成单一字符，必须写成%%

以上是dos 变量处理的通用格式，如果其中的 m 、n 为变量，那么这种情况就是变量嵌套了。

比如设变量 word为“abcdefghij”，变量 num 为“123456789 ”

%word:~4,1%为e，其中 4 可以从变量 num 中取值，即%num:~3,1%，写成组合形式如下：

%word:~%num:~3,1%,1% 经测试这种写法不能正确执行，写成%word:~(%num:~3,1%),1% 同样不行，那么，怎

么实现这种变量嵌套呢？这就必须结合命令嵌套。

什么是命令嵌套呢？简单的说，首先用一条 dos 命令生成一个字符串，而这个字符串是另一条 dos 命令，

用call 语句调用字符串将其执行，从而得到最终结果。

例：用call 语句实现命令嵌套

@echo off

set str1=aaa echo ok bbb

echo 初始字符串：%str1%

echo 生成命令字符串如下：

echo %str1:~4,7%

echo 运行命令字符串生成最终结果为：

call %str1:~4,7%

pause

运行显示：

初始字符串：aaa echo ok bbb

生成命令字符串如下：

echo ok

运行命令字符串生成最终结果为：

ok

请按任意键继续. . .

Filename: CMD 命令行高级教程.doc

Directory: C:\Documents and Settings\judy.su\Desktop

Template: C:\Documents and Settings\judy.su\Application

Data\Microsoft\Templates\Normal.dotm

Title:

Subject:

Author: judy.su

Keywords:

Comments:

Creation Date: 3/10/2009 5:41:00 PM

Change Number: 14

Last Saved On: 3/11/2009 4:10:00 PM

Last Saved By: judy.su

Total Editing Time: 142 Minutes

Last Printed On: 3/13/2009 2:49:00 PM

As of Last Complete Printing

Number of Pages: 49

Number of Words: 25,298 (approx.)

Number of Characters: 29,093 (approx.)