



计算机安全与维护

Windows系统安全维护与加固进阶



计算机维护的“进阶”和“基础”相比
体现在哪些方面？

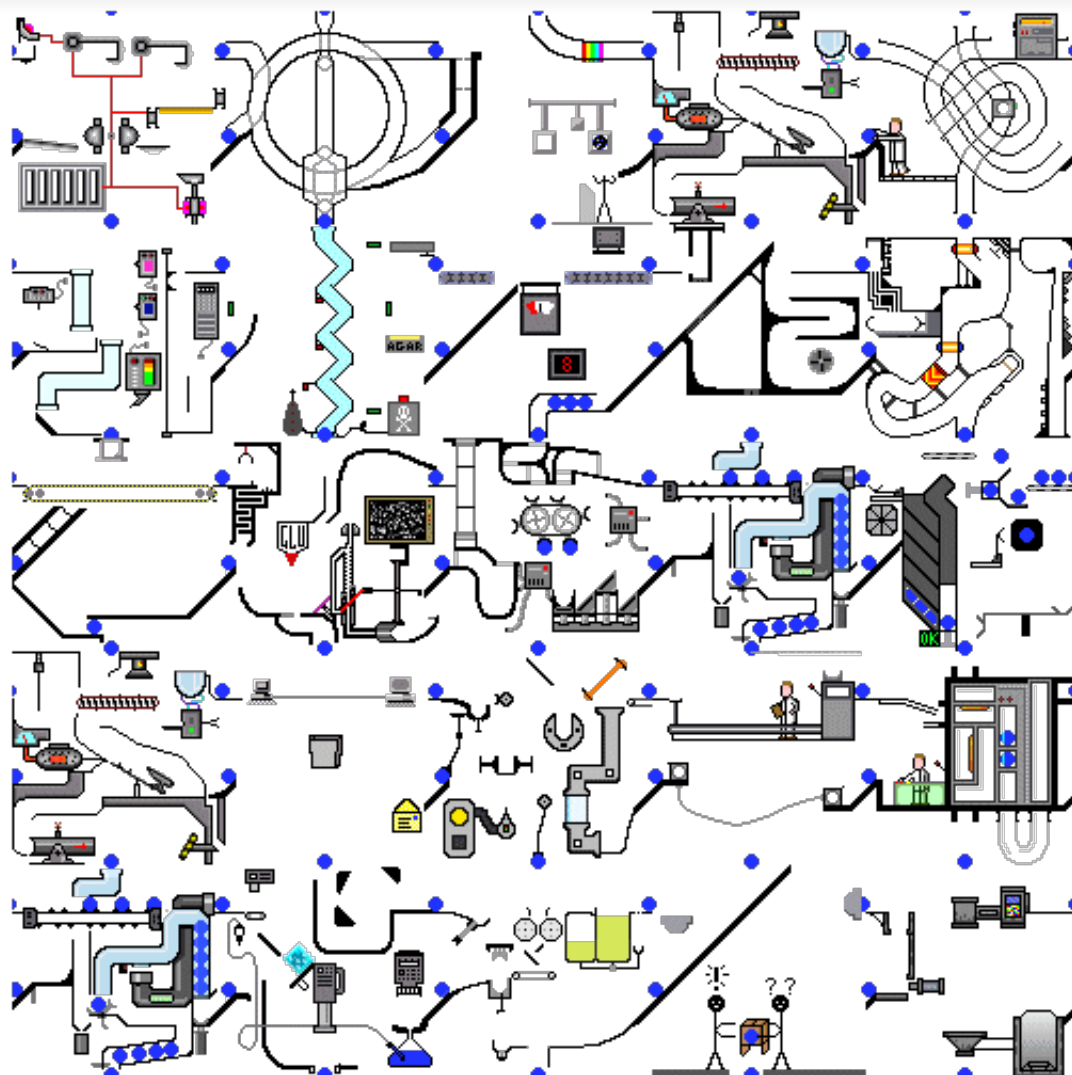


自动化的必要性

- 如果没有自动化，你怎么解决以下具体问题？
 - 1天之内，安装100台电脑？1000台？10000台？
 - 在所有新安装的电脑上
 - 安装配置好不同的用户帐号和密码
 - 安装配置不同的软件组合
 - 检测确认所有新电脑在新网络环境中可用



IT运维自动化——一图胜千言



中国传媒大学



自动化与说明书

- .exe 与 .txt的关系
- 没有人愿意阅读100页的使用说明书后才能开始使用一个新的IT设备或系统
- 自动化运维就是用可立即执行的程序代替操作手册



自动化的可行性

- 可编程
 - 特别是脚本编程
 - Windows 支持的主流脚本编程语言
 - 批处理脚本
 - PowerShell (基于.NET环境)
 - python



本章内容提要

- 批处理与自动化
- Windows的环境变量设置



CLI VS. GUI

	<i>CLI</i>	<i>GUI</i>
易用性	弱	强
可编程 (自动化)	强	弱

- CLI: Command Line Interface
- GUI: Graphics User Interface



批处理

- 批处理脚本
 - MS-DOS命令集合于一个文件
 - 每行一条命令
 - 文件扩展名: .bat / .cmd



.cmd和.bat的区别与联系

- 联系
 - 纯文本编码的脚本程序
- 区别
 - .cmd只支持win2000以上系统，.bat支持所有windows系统
 - .cmd允许使用的命令比.bat文件多
 - .cmd文件描述是：“windows nt命令行脚本”，.bat文件描述是：“ms dos批处理文件”



程序设计语言基本要素

中国传媒大学



基本语法

- 变量声明、定义与引用
- 运算符
- 表达式、代码注释
- 程序的控制结构
 - for / if / while / switch / goto
- 函数
- 程序的数据结构
- 文件操作



开发与运行环境

- 集成开发环境
 - 源代码编辑器
 - 程序的调试
 - 程序的运行
- 帮助体系
 - 帮助文档、手册



我们为什么需要集成开发环境

- 代码高亮、自动补全、函数间跳转、快速转到变量定义、快速查找到变量/函数的所有引用位置、一键显示帮助文档。。。



我们为什么要会用、勤用和善用“帮助体系”

- 系统和体系化学习一门程序设计语言的最权威资料
 - 官方文档和程序内置帮助文档
 - 官方的实例代码
- 不依赖于“上网”，我们可以无干扰的专注工作



简单批处理内部命令



echo

```
C:\Documents and Settings\lczh-kvm>echo /?  
显示信息，或将命令回显打开或关上。
```

```
ECHO [ON | OFF]
```

```
ECHO [message]
```

要显示当前回显设置，键入不带参数的 ECHO。

```
C:\Documents and Settings\lczh-kvm>echo  
ECHO 处于打开状态。
```



@

• 不回显@后面的命令

```
1 @echo off
2 SET st2Path=C:\tools\Sublime Text 2\sublime_text.exe
3
4 rem 为所有扩展名的文件添加右键菜单: Open with Sublime Text 2
5 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2" /t REG_SZ /v "" /d "Open with Sublime Text 2" /f
6 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2" /t REG_EXPAND_SZ /v "Icon" /d "%st2Path%,0" /f
7 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2\command" /t REG_SZ /v "" /d "%st2Path% \"%*1\"" /f
8
9 rem 为所有目录添加右键菜单: Open with Sublime Text 2
10 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_SZ /v "" /d "Open with Sublime Text 2" /f
11 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_EXPAND_SZ /v "Icon" /d "%st2Path%,0" /f
12 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2\command" /t REG_SZ /v "" /d "%st2Path% \"%*1\"" /f
13 pause
14
```



set

```
E:\>set /?
```

显示、设置或删除 `cmd.exe` 环境变量。

```
SET [variable=[string]]
```

variable 指定环境变量名。

string 指定要指派给变量的一系列字符串。

要显示当前环境变量，键入不带参数的 `SET`。

如果命令扩展被启用，`SET` 会如下改变：

可仅用一个变量激活 `SET` 命令，等号或值不显示所有前缀匹配
`SET` 命令已使用的名称的所有变量的值。例如：

```
SET P
```

会显示所有以字母 `P` 打头的变量

如果在当前环境中找不到该变量名称，`SET` 命令将把 `ERRORLEVEL` 设置成 `1`。

`SET` 命令不允许变量名含有等号。

请按任意键继续. . .



命令行『开关』

- 说法上等价于命令行『参数』
- 用正斜杠符号/表示开始一个『开关』参数
 - 大多数使用单个字母，也有使用多个字母
 - 其他操作系统的命令行参数通常使用减号：-
 - 命令行开关不区分大小写
 - 从其他操作系统移植到windows上的命令行程序依然保持其原有参数标识习惯：使用减号
 - 多个命令行开关之间大多数可以不加空格，极少数可以采用合并/，**没有强制规定，取决于命令行程序如何实现**



```
C:\Documents and Settings\lczh-kvm>goto /?
```

将 cmd.exe 导向到批处理程序中带标签的行。

```
GOTO label
```

label 指定批处理程序中用作标签的文字字符串。

标签必须单独一行，并且以冒号打头。

如果命令扩展名被启用，GOTO 会如下改变：

GOTO 命令现在接受目标标签 **:EOF**，这个标签将控制转移到当前批脚本文件的结尾。不定义就退出批脚本文件，这是一个容易的办法。有关能使该功能有用的 **CALL** 命令的扩展名描述，请键入 **CALL /?**。



rem

```
C:\Documents and Settings\lczh-kvm>rem /?  
在批处理文件或 CONFIG.SYS 里加上注解或说明。
```

```
REM [comment]
```

- 等价注释符号
 - ::
 - 执行效率更高，书写更方便
 - 即使设置echo on，::标记的行内容在被执行时不会回显，REM注释掉的内容会被回显
- 置于代码行尾时，需要使用 **& REM** 或 **& ::** 这种形式



pause

```
C:\Documents and Settings\lczh-kvm>pause /?  
暂停批处理程序，并显示以下消息：  
请按任意键继续. . .
```



call (1/3)

```
C:\Documents and Settings\lczh-kvm>call /?
```

从批处理程序调用另一个批处理程序。

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

batch-parameters 指定批处理程序所需的命令行信息。

如果命令扩展名被启用，**CALL** 会如下改变：

CALL 命令现在将卷标当作 **CALL** 的目标接受。语法是：

```
CALL:label arguments
```

一个新的批文件上下文由指定的参数所创建，控制在卷标被指定后传递到语句。您必须通过达到批脚本文件末两次来 "exit" 两次。第一次读到文件末时，控制会回到 **CALL** 语句的紧后面。第二次会退出批脚本。键入 **GOTO /?**，参看 **GOTO : EOF** 扩展名的描述，此描述允许您从一个批脚本返回。

另外，批脚本文本参数参照<%0、%1、等等>已如下改变：

批脚本里的 **%*** 指出所有的参数(如 %1 %2 %3 %4 %5 ...)



call (2/3)

批参数<%n>的替代已被增强。您可以使用以下语法：

- %~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 : 1 - 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:1 - 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。



call (3/3)

```

%~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 : 1 - 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:1 - 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。

```

在上面的例子中, %1 和 PATH 可以被其他有效数值替换。
%~ 语法被一个有效参数号码终止。%~ 修定符不能跟 %* 使用



start

```
C:\Documents and Settings\lczh-kvm>start /?
```

启动另一个窗口运行指定的程序或命令。

```
START ["title"] [/Dpath] [/I] [/MIN] [/MAX] [/SEPARATE : /SHARED]
      [/LOW : /NORMAL : /HIGH : /REALTIME : /ABOVENORMAL : /BELOWNORMAL]
      [/WAIT] [/B] [command/program]
      [parameters]
```

"title"	在窗口标题栏中显示的标题。
path	起始目录
B	在不创建新窗口的情况下开始应用程序。除非启动 ^C 处理，否则该应用程序会忽略 ^C 处理； ^Break 是唯一可以中断该应用程序的方式
I	新环境是传递给 cmd.exe 的原始环境，而不是当前环境
MIN	开始时窗口最小化
MAX	开始时窗口最大化
SEPARATE	在分开的空间内开始 16 位 Windows 程序
SHARED	在分共享的空间内开始 16 位 Windows 程序
LOW	在 IDLE 优先级类别开始应用程序
NORMAL	在 NORMAL 优先级类别开始应用程序
HIGH	在 HIGH 优先级类别开始应用程序
REALTIME	在 REALTIME 优先级类别开始应用程序
ABOVENORMAL	在 ABOVENORMAL 优先级类别开始应用程序

请按任意键继续. . .



choice (仅限Win 2003, Vista, Win7)

- choice 使用此命令可以让用户输入一个字符，从而运行不同的命令。使用时应该加/c:参数，c:后应写提示可输入的字符，之间无空格。它的返回码为1234.....



if

```
C:\Documents and Settings\lczh-kvm>if /?
```

执行批处理程序中的条件处理。

```
IF [NOT] ERRORLEVEL number command
```

```
IF [NOT] string1==string2 command
```

```
IF [NOT] EXIST filename command
```

NOT 指定只有条件为 **false** 的情况下，Windows XP 才应该执行该命令。

ERRORLEVEL number 如果最后运行的程序返回一个等于或大于指定数字的退出编码，指定条件为 **true**。

string1==string2 如果指定的文字字符串匹配，指定条件为 **true**。

EXIST filename 如果指定的文件名存在，指定条件为 **true**。

command 如果符合条件，指定要执行的命令。如果指定的条件为 **FALSE**，命令后可跟一个执行 **ELSE** 关键字后的命令的 **ELSE** 命令。

ELSE 子句必须在 **IF** 之后出现在同一行上。例如：

```
IF EXIST filename. <  
请按任意键继续. . .
```



for

```
C:\Users\huangwei>for /?
```

对一组文件中的每一个文件执行某个特定命令。

```
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 指定对每个文件执行的命令。
command-parameters
为特定命令指定参数或命令行开关。

在批处理程序中使用 **FOR** 命令时，指定变量请使用 **%variable** 而不要用 **%variable**。变量名称是区分大小写的，所以 **%i** 不同于 **%I**。

如果启用命令扩展，则会支持下列 **FOR** 命令的其他格式：

```
FOR /D %variable IN <set> DO command [command-parameters]
```

如果集中包含通配符，则指定与目录名匹配，而不与文件名匹配。

```
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** 后没有指定目录规范，则使用当前目录。如果集仅为一个单点 $\langle . \rangle$ 字符，
请按任意键继续。 . . .



for

则枚举该目录树。

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

该集表示以增量形式从开始到结束的一个数字序列。因此，<1,1,5>将产生序列 1 2 3 4 5，<5,-1,1>将产生序列<5 4 3 2 1>

```
FOR /F ["options"] %variable IN (file-set) DO command [command-parameters]
```

```
FOR /F ["options"] %variable IN ("string") DO command [command-parameters]
```

```
FOR /F ["options"] %variable IN ('command') DO command [command-parameters]
```

或者，如果有 usebackq 选项：

```
FOR /F ["options"] %variable IN (file-set) DO command [command-parameters]
```

```
FOR /F ["options"] %variable IN ("string") DO command [command-parameters]
```

```
FOR /F ["options"] %variable IN ('command') DO command [command-parameters]
```

fileset 为一个或多个文件名。继续到 fileset 中的下一个文件之前，每份文件都被打开、读取并经过处理。处理包括读取文件，将其分成一行行的文字，然后将每行解析成零或更多的符号。然后用已找到的符号字符串变量值调用 For 循环。

以默认方式，/F 通过每个文件的每一行中分开的第一个空白符号。跳过空白行。您可通过指定可选 "options" 参数替代默认解析操作。这个带引号的字符串包括一个或多个指定不同解析选项的关键字。这些关键字为：

请按任意键继续. . .



for

或多个指定不同解析选项的关键字。这些关键字为：

- eol=c** - 指一个行注释字符的结尾<就一个>
- skip=n** - 指在文件开始时忽略的行数。
- delims=xxx** - 指分隔符集。这个替换了空格和跳格键的默认分隔符集。
- tokens=x,y,m-n** - 指每行的哪一个符号被传递到每个迭代的 **for** 本身。这会导致额外变量名称的分配。**m-n** 格式为一个范围。通过 **nth** 符号指定 **nth**。如果符号字符串中的最后一个字符星号，那么额外的变量将在最后一个符号解析之后分配并接受行的保留文本。
- usebackq** - 指定新语法已在下类情况中使用：
在作为命令执行一个后引号的字符串并且一个单引号字符为文字字符串命令并允许在 **file-set** 中使用双引号扩起文件名称。

某些范例可能有助：

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do @echo %i %j %k
```

会分析 **myfile.txt** 中的每一行，忽略以分号打头的那些行，将每行中的第二个和第三个符号传递给 **for** 函数体，用逗号和/或空格分隔符号。请注意，此 **for** 函数体的语句引用 **%i** 来
请按任意键继续. . .



for

空格分隔符号。请注意，此 **for** 函数体的语句引用 **%i** 来获得第二个符号，引用 **%j** 来获得第三个符号，引用 **%k** 来获得第三个符号后的所有剩余符号。对于带有空格的文件名，您需要用双引号将文件名括起来。为了用这种方式来使用双引号，还需要使用 **usebackq** 选项，否则，双引号会被理解成是用作定义某个要分析的字符串的。

%i 在 **for** 语句中显式声明，**%j** 和 **%k** 是通过 **tokens=** 选项隐式声明的。可以通过 **tokens=** 一行指定最多 26 个符号，只要不试图声明一个高于字母 "z" 或 "Z" 的变量。请记住，**FOR** 变量是单一字母、分大小写和全局的变量；而且，不能同时使用超过 52 个。

还可以在相邻字符串上使用 **FOR /F** 分析逻辑，方法是，用单引号将括号之间的 **file-set** 括起来。这样，该字符串会被当作一个文件中的一个单一输入行进行解析。

最后，可以用 **FOR /F** 命令来分析命令的输出。方法是，将括号之间的 **file-set** 变成一个反括字符串。该字符串会被当作命令行，传递到一个子 **CMD.EXE**，其输出会被捕获到内存中，并被当作文件分析。如以下例子所示：

```
FOR /F "usebackq delims==" %i IN (<'set'>) DO Echo %i
```

请按任意键继续. . .



for

会枚举当前环境中的环境变量名称。

另外，FOR 变量参照的替换已被增强。您现在可以使用下列选项语法：

%~I	- 删除任何引号(">), 扩展 %I
%~fI	- 将 %I 扩展到一个完全合格的路径名
%~dI	- 仅将 %I 扩展到一个驱动器号
%~pI	- 仅将 %I 扩展到一个路径
%~nI	- 仅将 %I 扩展到一个文件名
%~xI	- 仅将 %I 扩展到一个文件扩展名
%~sI	- 扩展的路径只含有短名
%~aI	- 将 %I 扩展到文件的文件属性
%~tI	- 将 %I 扩展到文件的日期/时间
%~zI	- 将 %I 扩展到文件的大小
%~\$PATH:I	- 查找列在路径环境变量的目录, 并将 %I 扩展 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, 此组合键会扩展到 空字符串

可以组合修饰符来得到多重结果：

%~dpI - 仅将 %I 扩展到一个驱动器号和路径

请按任意键继续. . .



for

```
%~nI - 仅将 %I 扩展到一个文件名
%~xI - 仅将 %I 扩展到一个文件扩展名
%~sI - 扩展的路径只含有短名
%~aI - 将 %I 扩展到文件的文件属性
%~tI - 将 %I 扩展到文件的日期/时间
%~zI - 将 %I 扩展到文件的大小
%~$PATH:I - 查找列在路径环境变量的目录，并将 %I 扩展
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，此组合键会扩展到
空字符串
```

可以组合修饰符来得到多重结果：

```
%~dpI - 仅将 %I 扩展到一个驱动器号和路径
%~nxI - 仅将 %I 扩展到一个文件名和扩展名
%~fsI - 仅将 %I 扩展到一个带有短名的完整路径名
%~dp$PATH:I - 搜索列在路径环境变量的目录，并将 %I 扩展
到找到的第一个驱动器号和路径。
%~ftzaI - 将 %I 扩展到类似输出线路的 DIR
```

在以上例子中，%I 和 PATH 可用其他有效数值代替。%~ 语法用一个有效的 FOR 变量名终止。选取类似 %I 的大写变量名比较易读，而且避免与不分大小写的组合键混淆。



批处理进阶



区分内部命令与外部命令

• where

- 在当前目录和环境变量PATH所定义的目录里遍历查找文件
- Win 2003, vista 和win 7及后续版本有该程序
- WinXp没有内置该程序

内部命令

```
C:\Users\cuc>where cd
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where dir
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where echo
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where goto
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where if
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where for
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where set
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where start
信息: 用提供的模式无法找到文件。

C:\Users\cuc>where type
信息: 用提供的模式无法找到文件。
```

外部命令 (程序)

```
C:\Users\cuc>where where
C:\Windows\System32\where.exe

C:\Users\cuc>where cmd
C:\Windows\System32\cmd.exe

C:\Users\cuc>where regedit
C:\Windows\regedit.exe

C:\Users\cuc>where notepad
C:\Windows\System32\notepad.exe
C:\Windows\notepad.exe

C:\Users\cuc>where find
C:\Windows\System32\find.exe

C:\Users\cuc>where tasklist
C:\Windows\System32\tasklist.exe

C:\Users\cuc>where taskkill
C:\Windows\System32\taskkill.exe

C:\Users\cuc>where more
C:\Windows\System32\more.com
```



解释型脚本与逐『行』执行

• 实例一

```
@echo off
```

```
ping -n 10 127.0.0.1 & REM wait
```

```
echo A
```

- 在脚本执行过程中（未执行到echo A这一行代码前），修改脚本源代码中的A为B，则最终脚本输出的是B，而不是A

• 实例二

```
@echo off
```

```
for /L %%i in (1,1,10) do (
```

```
    echo B
```

```
    ping -n 2 127.0.0.1 >nul & REM wait
```

```
)
```

- 在脚本执行过程中，修改脚本源代码中的A为B，则最终脚本输出依然是A



如何在批处理中使用参数

- %1 ... %9
- 超过9个参数时，使用shift指令来移动



使用组合命令（复合语句）

- 第一条命令 & 第二条命令 [& 第三条命令…]
 - 不管命令是否执行成功，顺序遍历执行
- 第一条命令 && 第二条命令 [&& 第三条命令…]
 - 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
- 第一条命令 || 第二条命令 [|| 第三条命令…]
 - 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



管道命令的使用 (1/2)

- 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
- >、>> 输出重定向命令
 - 将一条命令或某个程序输出结果的重定向到特定文件中,>与>>的区别在于,>会清除调原有文件中的内容后写入指定文件,而>>只会追加内容到指定文件中,而不会改动其中的内容。



管道命令的使用 (2/2)

- `<`、`>&`、`<&`
 - `<` 从文件中而不是从键盘中读入命令输入
 - `>&h` 将一个句柄的输出写入到另一个句柄的输入中
 - `<&h` 从一个句柄读取输入并将其写入到另一个句柄输出中
 - 句柄h的取值范围：
 - 0 标准输入
 - 1 标准输出，默认控制台程序的输出就是标准输出
 - 2 标准错误输出
 - 其他合法句柄表示方法，例如：文件路径



重定向实例

- `dir non-exist >info.log`
- `dir non-exist >info.log 2>&1`
- `dir non-exist 2>err.log 1>info.log`



特殊字符 (1/3)

- @ > >> |
- ^
 - ^ 是对特殊符号 >、<、& 的前导字符。在命令中它将以上的3个符号的特殊功能去掉仅仅只把他们当成符号而不使用他们的特殊意义，即：转义符
 - 行尾使用^时，下一行的代码会被当作是当前行代码的延续
- "
 - 允许在字符串中包含空格



特殊字符 (2/3)

- ,
 - ,符号相当于空格。在某些特殊的情况下可以用,来代替空格使用
- ;
 - 当命令相同的时候可以将不同的目标用;隔离开来但执行效果不变。如执行过程中发生错误则只返回错误报告但程序还是会继续执行
- %
 - 批处理中使用%需要转义,转义方法是%%



特殊字符 (3/3)

- !
 - 在setlocal enabledelayedexpansion语句之后，使用!!引用变量时，变量设置为运行时赋值
 - 用于逻辑非计算符号
- nul
 - windows上的『空』设备，俗称数据黑洞。相当于UNIX系统中的/dev/null
 - 输出被重定向到null就相当于输出不回显



设置局部(环境)变量-setlocal ... endlocal

- 变量设置只在setlocal 和 endlocal之间有效
 - endlocal之后即无法访问在setlocal之后定义的局部变量值



实例



命令行操作进阶

- `tasklist | find "cmd"`
- `tasklist | more`
- `dir | find ".bat"`
- `dir /s /b notepad.exe`



在Windows资源管理器添加自定义右键菜单

```
1 @echo off
2 SET st2Path=C:\tools\Sublime Text 2\sublime_text.exe
3
4 rem 为所有扩展名的文件添加右键菜单: Open with Sublime Text 2
5 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2" /t REG_SZ /v "" /d "Open with Sublime Text 2" /f
6 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2" /t REG_EXPAND_SZ /v "Icon" /d "%st2Path%,0" /f
7 @reg add "HKEY_CLASSES_ROOT*\shell\Open with Sublime Text 2\command" /t REG_SZ /v "" /d "%st2Path% \"%*1\"" /f
8
9 rem 为所有目录添加右键菜单: Open with Sublime Text 2
10 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_SZ /v "" /d "Open with Sublime Text 2" /f
11 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_EXPAND_SZ /v "Icon" /d "%st2Path%,0" /f
12 @reg add "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2\command" /t REG_SZ /v "" /d "%st2Path% \"%*1\"" /f
13 pause
T3 b9n26
T5 GL6B sqq "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_SZ /v "" /d "Open with Sublime Text 2" /f
T7 GL6B sqq "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2" /t REG_EXPAND_SZ /v "Icon" /d "%st2Path%,0" /f
T9 GL6B sqq "HKEY_CLASSES_ROOT\Folder\shell\Open with Sublime Text 2\command" /t REG_SZ /v "" /d "%st2Path% \"%*1\"" /f
```



学习资源

- <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/batch.mspix?mfr=true>
- MSDN论坛: <http://social.msdn.microsoft.com/Forums/zh-CN/home>
- Windows CMD命令分类索引: <http://ss64.com/nt/commands.html>
- Windows CMD语法: <http://ss64.com/nt/syntax.html>
- 批处理编程实例与技巧: <http://www.robvanderwoude.com/battech.php>
- Windows命令行指南: http://en.wikibooks.org/wiki/Guide_to_Windows_commands



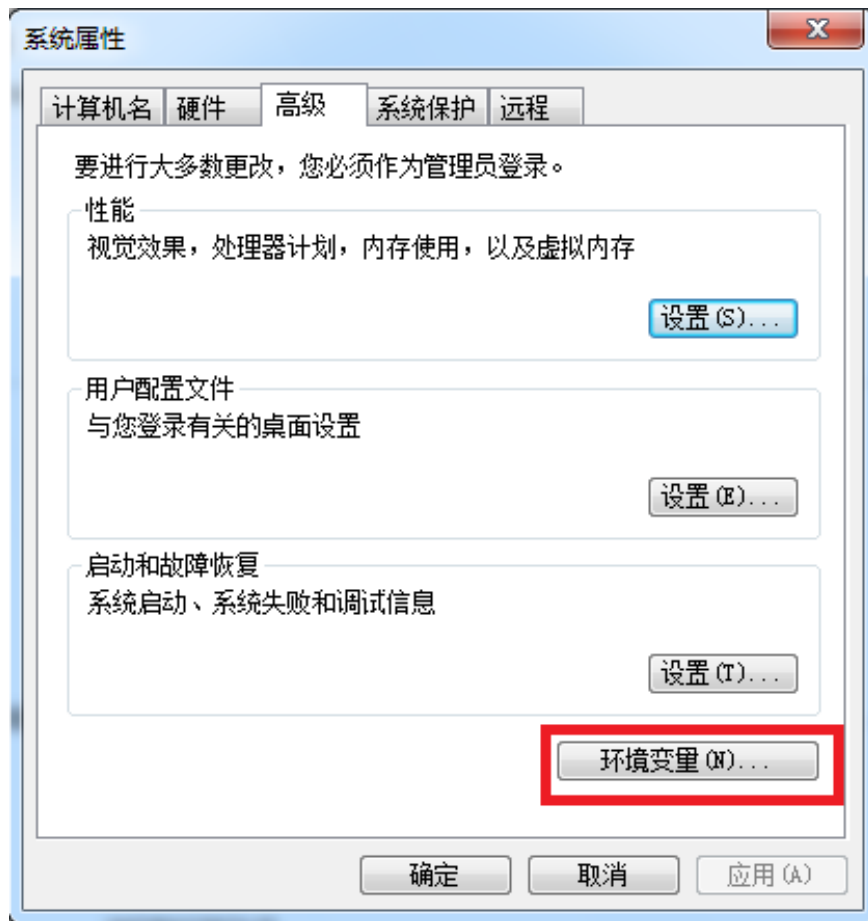
本章内容提要

- 批处理与自动化
- Windows的环境变量设置



Windows的环境变量设置

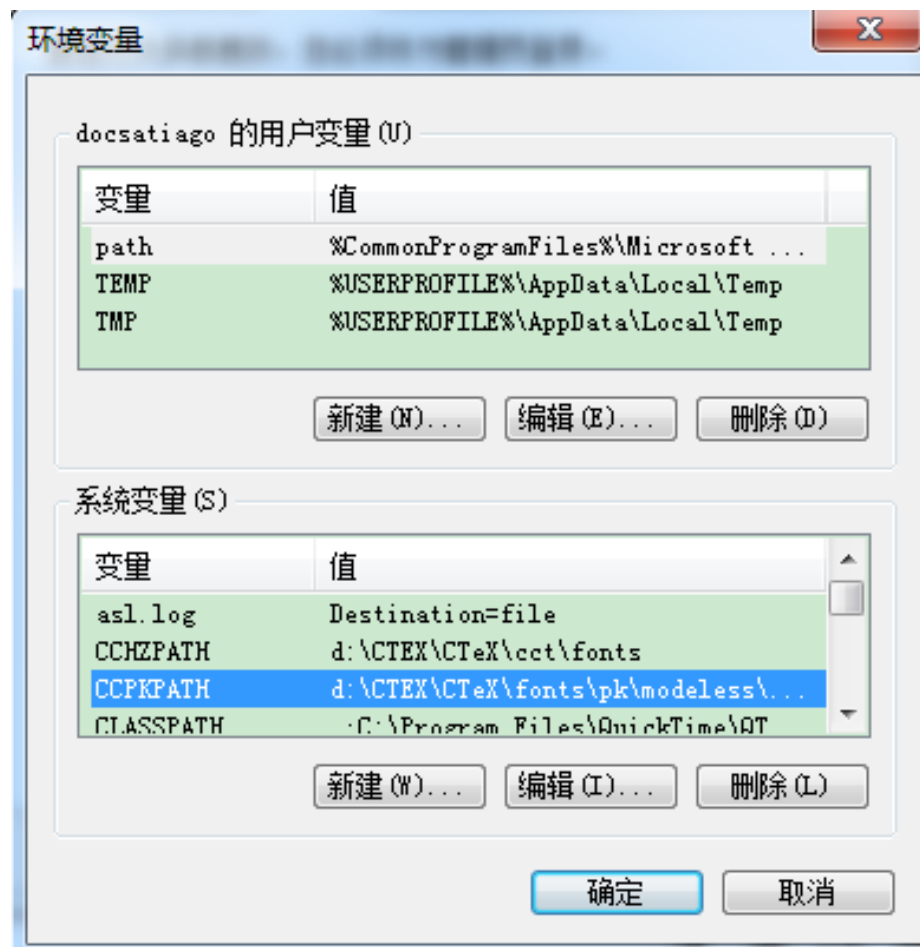
- Windows 电脑“系统属性”





Windows的环境变量设置

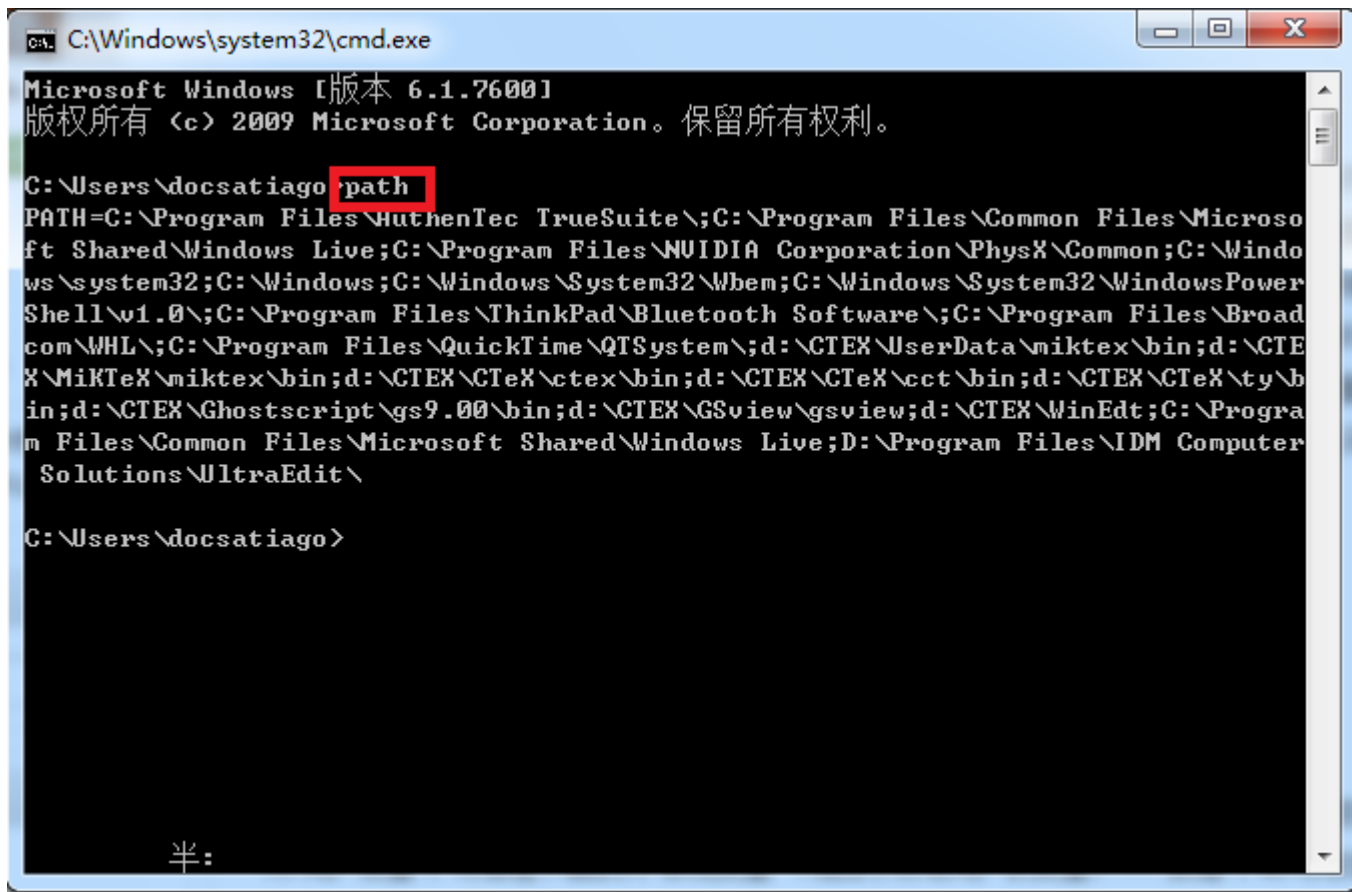
- 环境变量设置界面





Windows的环境变量设置

- 命令查看环境变量



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\docsatiago> path
PATH=C:\Program Files\HuthenTec TrueSuite\;C:\Program Files\Common Files\Microso
ft Shared\Windows Live;C:\Program Files\NVIDIA Corporation\PhysX\Common;C:\Windo
ws\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPower
Shell\v1.0\;C:\Program Files\ThinkPad\Bluetooth Software\;C:\Program Files\Broad
com\WHL\;C:\Program Files\QuickTime\QTSystem\;d:\CTEX\UserData\miktex\bin;d:\CTE
X\MiKTeX\miktex\bin;d:\CTEX\CTeX\ctex\bin;d:\CTEX\CTeX\cct\bin;d:\CTEX\CTeX\tyb
in;d:\CTEX\Ghostscript\gs9.00\bin;d:\CTEX\GSview\gsview;d:\CTEX\WinEdt;C:\Progra
m Files\Common Files\Microsoft Shared\Windows Live;D:\Program Files\IDM Computer
Solutions\UltraEdit\

C:\Users\docsatiago>
```



Windows的环境变量设置

- echo %变量名%形式显示环境变量

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\docsatiago>echo %classpath%
.;C:\Program Files\QuickTime\QTSystem\QTJava.zip

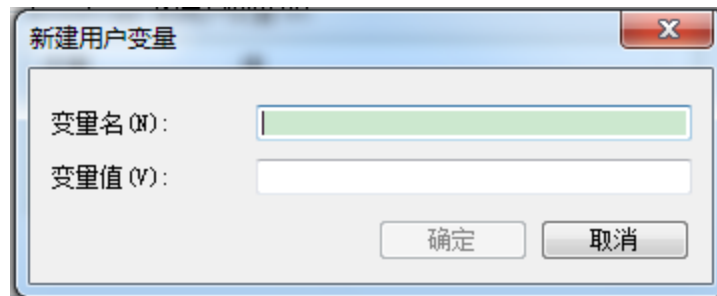
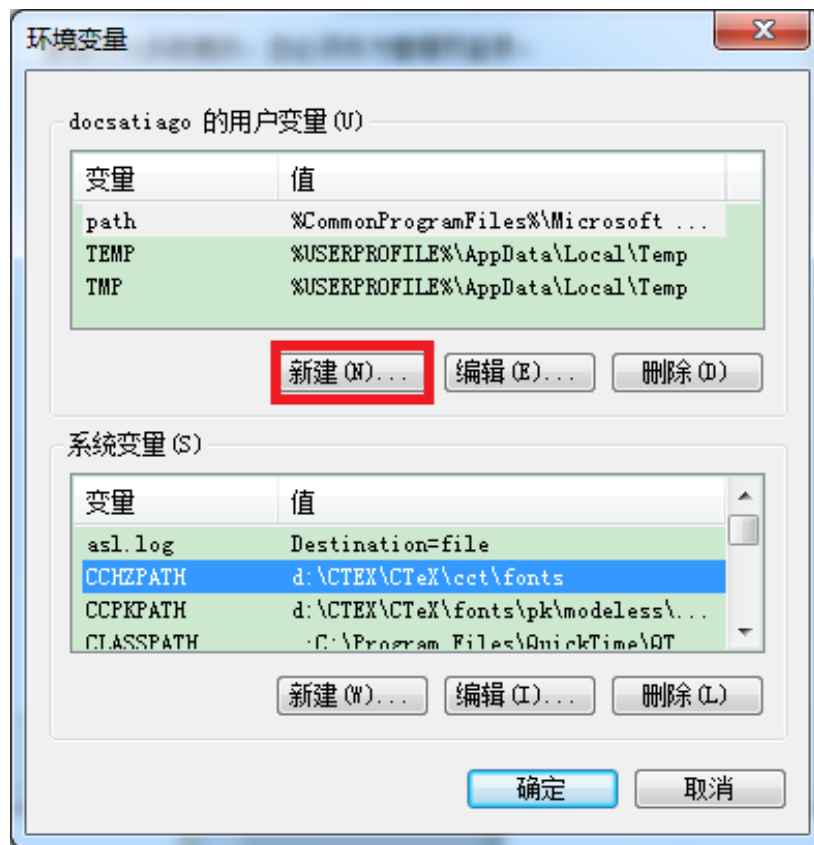
C:\Users\docsatiago>echo %path%
C:\Program Files\AuthenTec TrueSuite\;C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Program Files\NVIDIA Corporation\PhysX\Common;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\ThinkPad\Bluetooth Software\;C:\Program Files\Broadcom\WMHL\;C:\Program Files\QuickTime\QTSystem\;d:\CTEX\UserData\miktex\bin;d:\CTEX\MikTeX\miktex\bin;d:\CTEX\CTEX\ctex\bin;d:\CTEX\CTEX\cct\bin;d:\CTEX\CTEX\ty\bin;d:\CTEX\Ghostscript\gs9.00\bin;d:\CTEX\GSview\gsview;d:\CTEX\WinEdt;C:\Program Files\Common Files\Microsoft Shared\Windows Live;D:\Program Files\IDM Computer Solutions\UltraEdit\

C:\Users\docsatiago>
```




Windows的环境变量设置

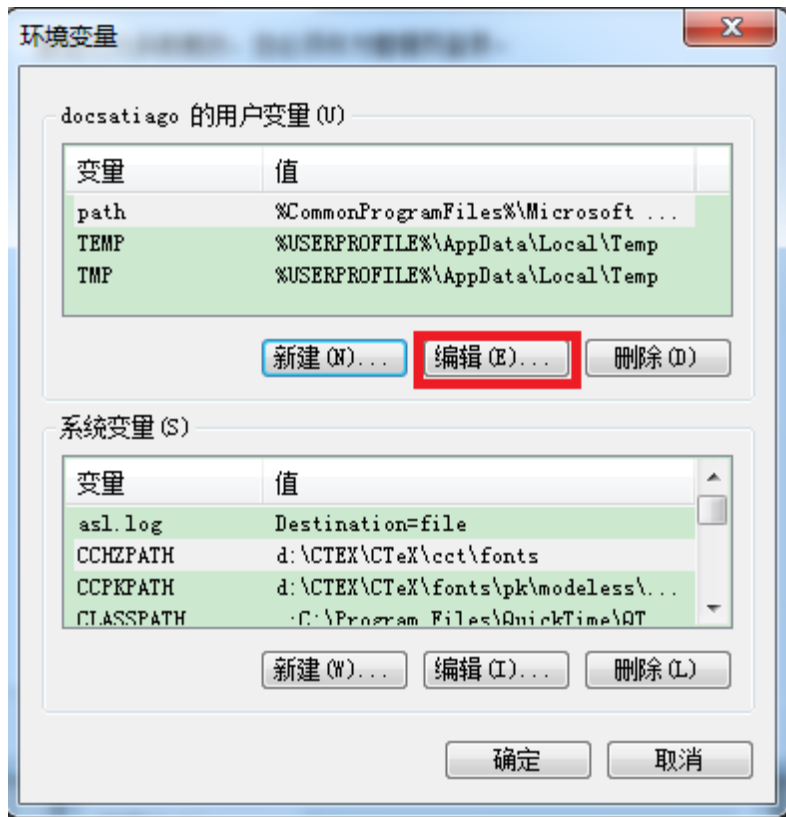
- 新建环境变量





Windows的环境变量设置

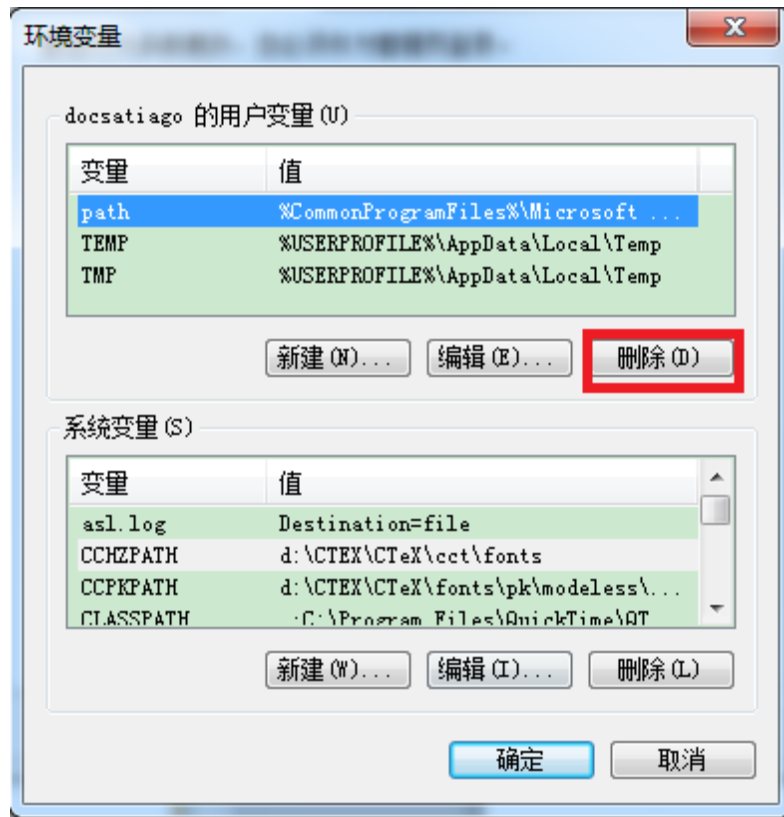
- 编辑环境变量





Windows的环境变量设置

- 删除环境变量





Windows的环境变量设置

- 系统变量

- 与windows操作系统包括网络状况有关，由操作系统定义。Administrators组的用户可以添加添加、修改或删除，
- 系统变量的作用域是全局的，对所有用户生效

- 用户变量

- 由操作系统、某些应用程序以及用户建立，任何用户都可以添加、修改或删除
- 用户变量的作用域是当前用户，只对当前用户生效



Windows的环境变量设置

- %ALLUSERSPROFILE% : 列出所有用户Profile文件位置。
- %APPDATA% : 列出应用程序数据的默认存放位置。
- %CD% : 列出当前目录。
- %CLIENTNAME% : 列出联接到终端服务会话时客户端的NETBIOS名。
- %CMDCMDLINE% : 列出启动当前cmd.exe所使用的命令行。



Windows的环境变量设置

- %CMDEXTVERSION%：命令出当前命令处理程序扩展版本号。
- %CommonProgramFiles%：列出了常用文件的文件夹路径。
- %COMPUTERNAME%：列出了计算机名。
- %COMSPEC%：列出了可执行命令外壳（命令处理程序）的路径。
- %DATE%：列出当前日期。



Windows的环境变量设置

- %ERRORLEVEL%：列出了最近使用的命令的错误代码。
- %HOMEDRIVE%：列出与用户主目录所在的驱动器盘符。
- %HOMEPATH%：列出用户主目录的完整路径。
- %HOMESHARE%：列出用户共享主目录的网络路径。



Windows的环境变量设置

- %LOGONSERVER%：列出有效的当前登录会话的域名控制器名。
- %NUMBER_OF_PROCESSORS%：列出了计算机安装的处理器的数。
- %OS%：列出操作系统的名字。(Windows XP 和 Windows 2000 列为 Windows_NT.)
- %Path%：列出了可执行文件的搜索路径。
- %PATHEXT%：列出操作系统认为可被执行的文件扩展名。



Windows的环境变量设置

- %PROCESSOR_ARCHITECTURE%：列出了处理器的芯片架构。
- %PROCESSOR_IDENTIFIER%：列出了处理器的描述。
- %PROCESSOR_LEVEL%：列出了计算机的处理器型号。
- %PROCESSOR_REVISION%：列出了处理器的修订号。



Windows的环境变量设置

- %ProgramFiles%：列出了Program Files文件夹的路径。
- %PROMPT%：列出了当前命令解释器的命令提示设置。
- %RANDOM%：列出界于0 和 32767之间的随机十进制数。



Windows的环境变量设置

- %SESSIONNAME%：列出连接到终端服务会话时的连接和会话名。
- %SYSTEMDRIVE%：列出了Windows启动目录所在驱动器。
- %SYSTEMROOT%：列出了Windows启动目录的位置。
- %TEMP% and %TMP%：列出了当前登录的用户可用应用程序的默认临时目录。
- %TIME%：列出当前时间。



Windows的环境变量设置

- %USERDOMAIN%：列出了包含用户帐号的域的名字。
- %USERNAME%：列出当前登录的用户的名字。
- %USERPROFILE%：列出当前用户Profile文件位置。
- %WINDIR%：列出操作系统目录的位置。



课内实验

- 实验一 安装并使用高级文本编辑器Sublime Text 3
 - 系统文件管理器右键菜单添加“edit with sublime text 3”
 - 代码自动补全（默认快捷键修改）
 - Preferences - Key Bindings - User
 - 在[]之间添加如下配置代码
 - { "keys": ["alt+/"], "command": "auto_complete" },
 - 宏录制与重放，完成琐碎重复编辑工作
 - 语法高亮设置



课内实验

• 实验二（批处理脚本编程）

- 批量删除某目录及其所有子目录下所有扩展名为.bak的文件
- 批量拷贝C:\Users\huangwei及其所有子目录下所有扩展名为.txt的文件到E:\Test下
 - E:\Test默认不存在
 - 拷贝后的文件名不变
 - 保持拷贝源文件的目录结构不变
 - 例如：C:\Users\huangwei\test\1\1.txt拷贝到目的文件夹后的绝对路径为：
E:\Test\Users\huangwei\test\1\1.txt



课内实验

- 实验三 windows环境变量查看和设置
 - 在用户配置界面查看环境变量和设置环境变量
 - 在命令行界面查看环境变量和设置环境变量