CSNB113: System Administration
-
10th Topic:
Shell Scripts – Windows

Windows needs scripting, too

Scripting on Windows is not so much different.

On Windows, you also have an environment:

- ComSpec specifies the command interpreter
- PATH specifies the locations to search for commands
- Prompt specifies the command prompt
- TEMP specifies the directories for temporary files

Most relevant difference to *nix: The 'replacement' indicator is %variable%; that means enclosing the variable in percent-marks. The typical prompt on Windows is '>', compared to the '$' on *nix.

Therefore, you obtain your environment like this:

```
echo %PATH%
echo %ComSpec%
```

`echo %windir%` returns the directory into which you have installed Windows, the ***System Directory***.

Practicalities

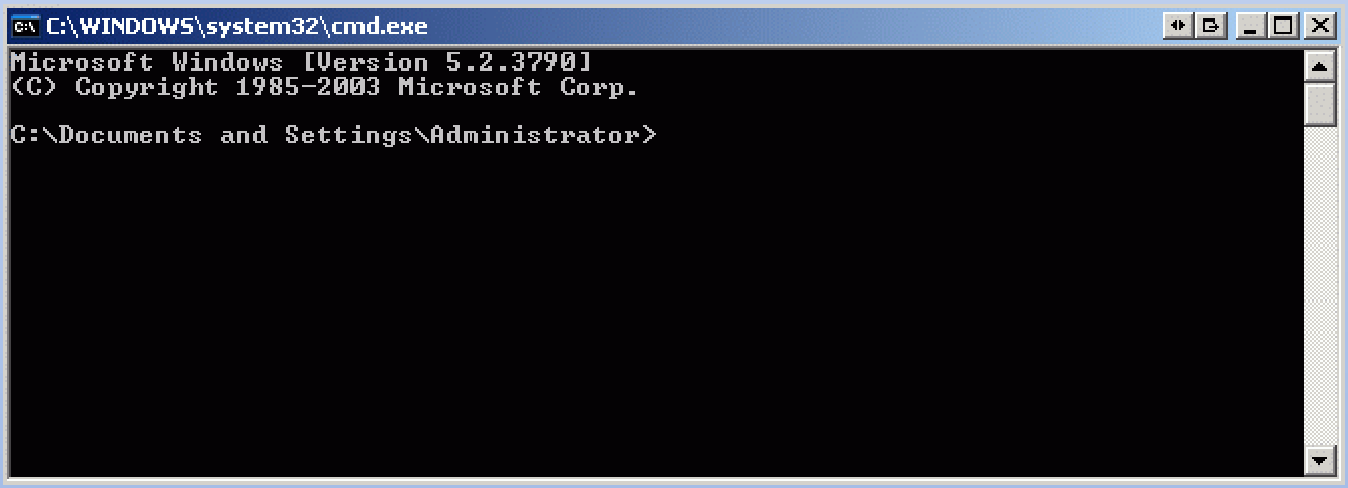
On Windows, the Command Prompt is not that obvious to access.

Start → All Programs → Accessories → Command Prompt

or:

Start → Run ... "cmd" → OK

Then a window will open and show the command prompt:



```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>
```

Commands - different

The basic commands are slightly different:
[windows command: explanation (unix command)]

DIR: list of files and directories (ls)

DATE: sets / displays the date

TIME: sets / displays the time

CLS: clear the screen (clear)

DIR *.* /S: displays recursively lower directories (ls -R)

TYPE: show content (cat)

DOSKEY /h: shows history of commands (history)

EDIT {filename}

ATTRIB: changes permissions (chmod)

SUBST: mount a drive to a directory (mount)

VER: shows the version of the operating system (uname)

DEL: deletes a files (rm)

CHDIR: shows current directory (pwd)

{command} /?: shows the available help (man)

Commands - similar

Some commands are similar, though:

CD: changes into a directory – difference: CD does not get the user back to the home directory

MKDIR: creates a directory (mkdir)

RMDIR: removes directory (rmdir)

MORE: shows one page at a time (more)

EXIT: exits the current session (exit)

ECHO: echos some text (echo)

FDISK: partitions a drive (fdisk)

HOSTNAME: shows the actual hostname (hostname)

REBOOT: reboots the system (reboot)

All identical are ***shell redirections***: > < >> >> and ***pipes*** '|'

Scripts

We write scripts like what we did for *nix:

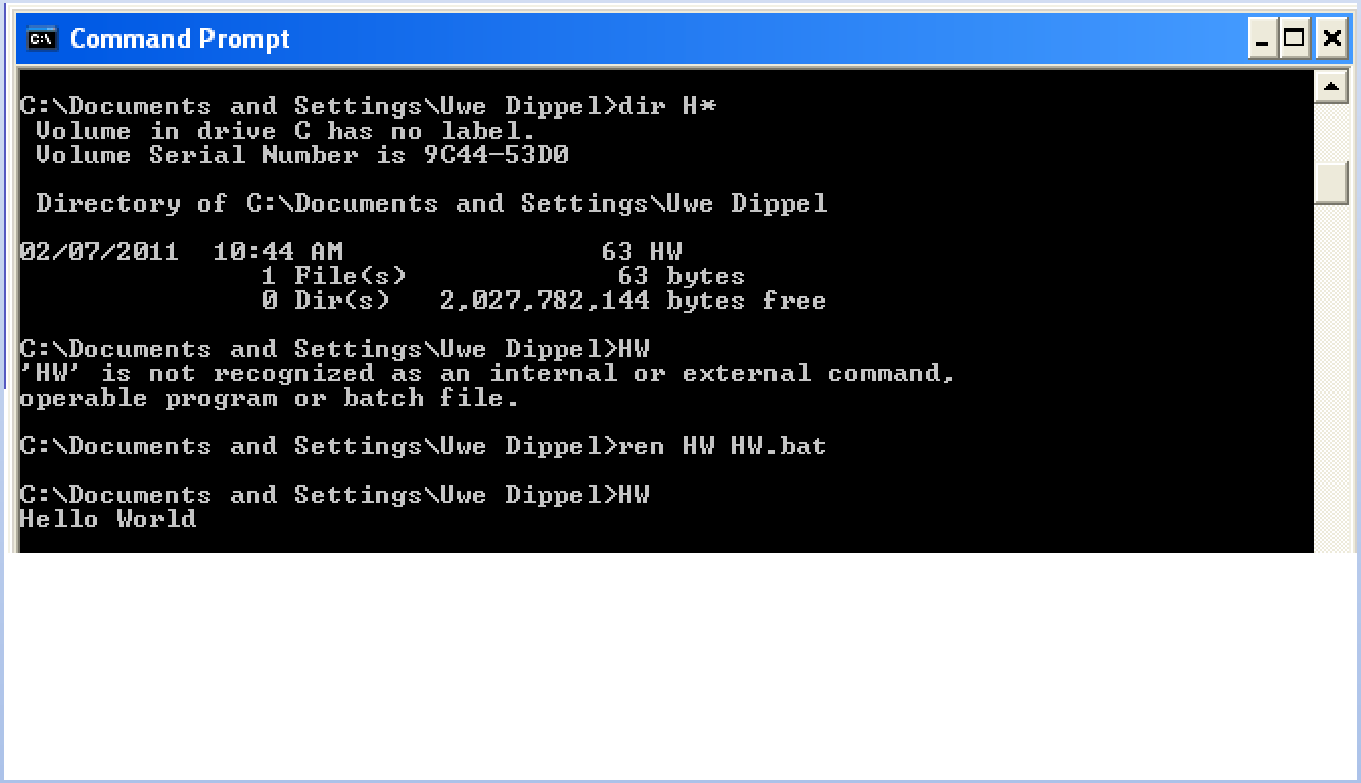
```
@echo off
set msg1=Hello
set msg2=World
echo %msg1% %msg2%
```

This of course prints `Hello World`

Some remarks on different behaviours:

In Unix, the execution is done by changing the permissions. Since Windows has no clear e'x'ecute permissions, this is done through the extension: A file needs a .bat, .exe or .com extension to be viewed as ***runnable***. For scripts, we use .bat

Examples - .bat



```
C:\Documents and Settings\Uwe Dippel>dir H*
 Volume in drive C has no label.
 Volume Serial Number is 9C44-53D0

 Directory of C:\Documents and Settings\Uwe Dippel

02/07/2011  10:44 AM                    63 HW
               1 File(s)             63 bytes
               0 Dir(s)   2,027,782,144 bytes free

C:\Documents and Settings\Uwe Dippel>HW
'HW' is not recognized as an internal or external command,
operable program or batch file.

C:\Documents and Settings\Uwe Dippel>ren HW HW.bat

C:\Documents and Settings\Uwe Dippel>HW
Hello World
```

Scripting Decisions

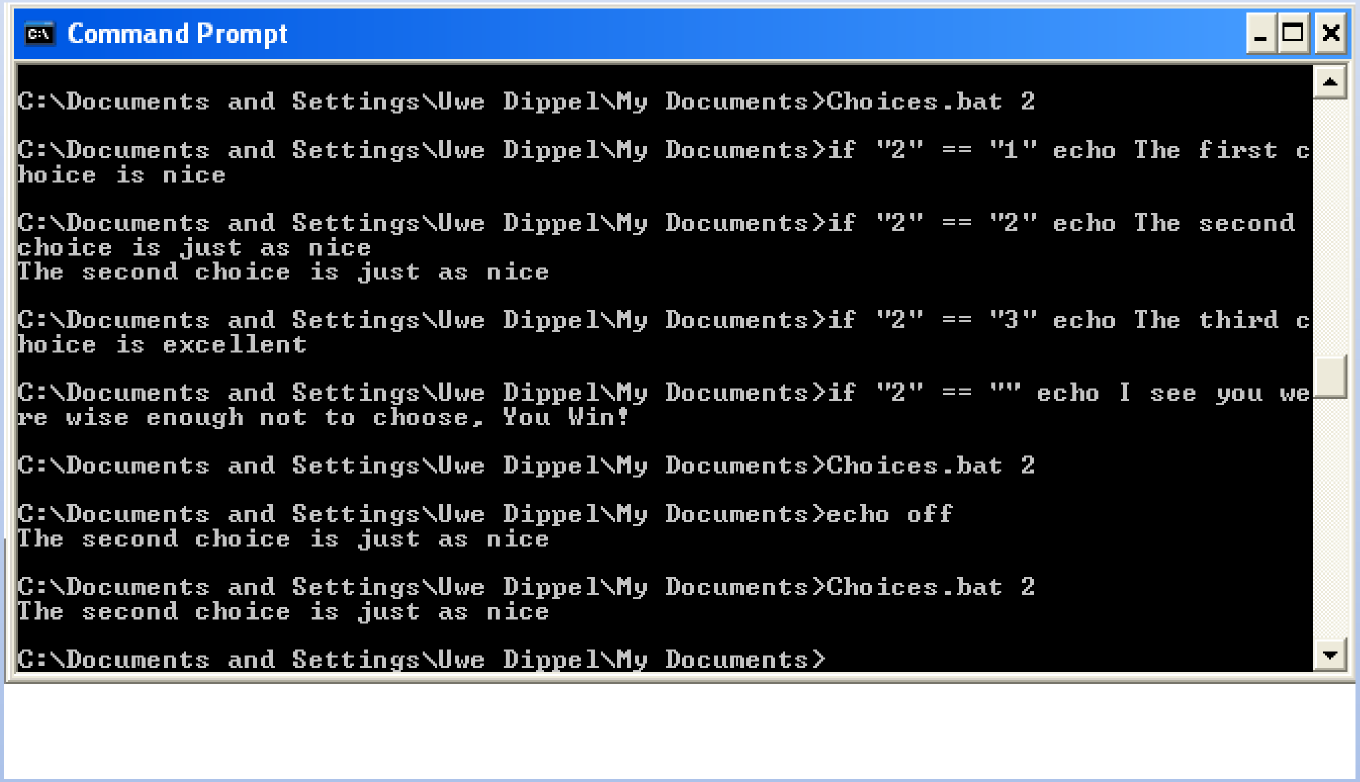
We write scripts like what we did for *nix:

```
@echo off
if "%1"=="1" echo The first choice is nice
if "%1"=="2" echo The second choice is just as nice
if "%1"=="3" echo The third choice is excellent
if "%1"==""  echo I see you were wise enough not to choose, You Win!
```

This shows that %1 is similar to $1 in *nix: it is the first argument.
The program works identical to the one that we had on *nix in an earlier chapter.

Some remarks on different behaviours:
In *nix, only the output is displayed, in Windows all lines are shown, including all the processes. If you only want the output, you need to 'echo off' the calculations and other inner workings. The '@' in front of a line suppresses the echo. So when you write 'echo off', all subsequent lines will not echo, but the first line, 'echo off', will echo still. Therefore we need the '@' in front of that line, so that the line itself is not echo-ed.

Examples – ECHO OFF


Command Prompt

```
C:\Documents and Settings\Uwe Dippel\My Documents>Choices.bat 2

C:\Documents and Settings\Uwe Dippel\My Documents>if "2" == "1" echo The first c
hoice is nice

C:\Documents and Settings\Uwe Dippel\My Documents>if "2" == "2" echo The second
choice is just as nice
The second choice is just as nice

C:\Documents and Settings\Uwe Dippel\My Documents>if "2" == "3" echo The third c
hoice is excellent

C:\Documents and Settings\Uwe Dippel\My Documents>if "2" == "" echo I see you we
re wise enough not to choose, You Win!

C:\Documents and Settings\Uwe Dippel\My Documents>Choices.bat 2

C:\Documents and Settings\Uwe Dippel\My Documents>echo off
The second choice is just as nice

C:\Documents and Settings\Uwe Dippel\My Documents>Choices.bat 2
The second choice is just as nice

C:\Documents and Settings\Uwe Dippel\My Documents>
```

Loops: For

There is a 'for'-loop construction:

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

```
@echo off
FOR /L %%i IN (0 5 30) DO (
   echo the next steps of 5 is:
   echo %%i
   echo ***
)
```

What does this code do?
(The /L-option turns on numeric mode, so that numerals are processed instead of numbers)

And this one?

```
@echo off
FOR %%f IN (*.txt) DO type %%f
```

Loops: counter

This is a very weird code:

```
@echo off
rem A Bang Counter
set n=%2
set i=
: loop
set i=%i%%1
  echo Bang!
  if %i%==%n% goto end
goto loop
:end
```

What does this code do?

References

- http://www.google.com.my/
  search for 'shell script'
- http://www.csie.ntu.edu.tw/~r92092/ref/win32/win32scripting.html